March 20–23, 2024
Portland, OR, USA

Association for Computing Machinery

Advancing Computing as a Science & Profession



SIGCSE 2024

Volume 2 of 2

Proceedings of the 55th ACM Technical Symposium on Computer Science Education

Sponsor:

ACM SIGCSE

Symposium Co-Chairs:

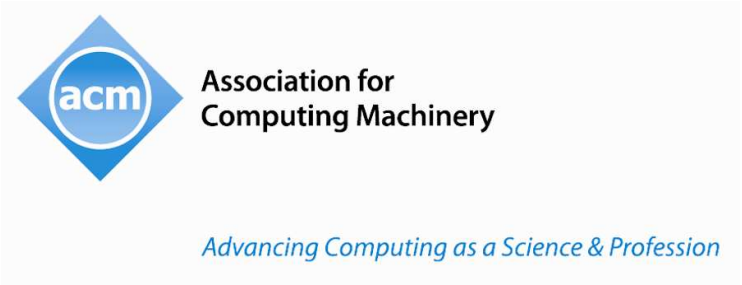
Ben Stephenson (University of Calgary, Canada)
Jeffrey A. Stone (Penn State University, USA)

Program Co-Chairs:

Lina Battestilli (North Carolina State University, USA)
Samuel A. Rebelsky (Grinnell College, USA)
Libby Shoop (Macalester College, USA)

Hybrid Experience Co-Chairs:

Delaram Yazdansepas (Loyola Marymount University, USA)
Oluwakemi Ola (University of British Columbia, Canada)
Kristin Stephens-Martinez (Duke University, USA)



The Association for Computing Machinery
1601 Broadway, 10th Floor
New York, NY 10019-7434

Copyright © 2024 by the Association for Computing Machinery, Inc. (ACM).
Permission to make digital or hard copies of portions of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission to republish from: permissions@acm.org or Fax +1 (212) 869-0481.

For other copying of articles that carry a code at the bottom of the first or last page, copying is permitted provided that the per-copy fee indicated in the code is paid through www.copyright.com.

ISBN: 979-8-4007-0424-6

Additional copies may be ordered prepaid from:

ACM Order Department
PO Box 30777
New York, NY 10087-0777, USA

Phone: 1-800-342-6626 (USA and Canada)
+1-212-626-0500 (Global)
Fax: +1-212-944-1318
E-mail: acmhelp@acm.org
Hours of Operation: 8:30 am – 4:30 pm ET

Printed in the USA

Table of Contents

SIGCSE TS 2024 Symposium Organization xlv

SIGCSE TS 2024 Reviewers l

SIGCSE TS 2024 Sponsors & Supporters lxii

Volume 1

Keynote Talks

- **Inspiring a Galaxy of New Innovators at the Intersection of Art & Science** 1
  Rachel Rose *(Industrial Light & Magic)*
- **Guiding Students Along Their Academic Path: Key Aspects of Learning That Impact All Learners** 3
  Todd Zakrajsek *(University of North Carolina at Chapel Hill)*

Research / Technical Papers

- **Hearing Iterative and Recursive Behavior: Sonification Improves Student Understanding** 4
  Joel Adams *(Dept. of Computer Science, Calvin University)*,
  Hayworth Anderson *(Dept. of Computer Science, Calvin University)*
- **Broadening Participation in Adult Education: A Literature Review of Computer Science Education** 11
  Friday Joseph Agbo *(School of Computing and Information Sciences, Willamette University)*
- **Do Behavioral Factors Influence the Extent to which Students Engage with Formative Practice Opportunities?** 18
  Ashish Aggarwal *(Department of Engineering Education, University of Florida)*,
  Manas Adepu *(Department of Comp. & Info. Science and Engineering, University of Florida)*,
  Alex Garcia-Marin *(Department of Comp. & Info. Science and Engineering, University of Florida)*,
  Christina Gardner-Mccune *(Department of Comp. & Info. Science and Engineering, University of Florida)*
- **Can Lexical Sophistication and Cohesion Automatically Differentiate Student Engagement in Socio-technical Platforms?** 25
  Mahir Akgun *(Pennsylvania State University)*, Priya Sharma *(Pennsylvania State University)*,
  Qiyuan Li *(Boston University)*
- **How We Manage an Army of Teaching Assistants: Experience Report on Scaling a CS1 Course** 32
  Ildar Akhmetov *(Northeastern University)*, Sadaf Ahmed *(University of Alberta)*,
  Kezziah Ayuno *(University of Alberta)*
- **NaijaCoder: Participatory Design for Early Algorithms Education in the Global South** 39
  Daniel Alabi *(Columbia University)*,
  Atinuke Adegbile *(Global Integrated Education Volunteers Association (GIEVA)*,
  Lekan Afuye *(Cornell University)*, Philip Abel *(Twilio, Inc.)*,
  Alida Monaco *(ICF International)*
- **Improving Knowledge of CS1 Terminology Through a Peer Reviewed Translation Activity: Results and Feedback** 46
  Suad Alaofi *(University College Dublin)*, Seán Russell *(University College Dublin)*
- **Can Language Models Employ the Socratic Method? Experiments with Code Debugging** 53
  Erfan Al-Hossami *(Computer Science, University of North Carolina at Charlotte)*,
  Razvan Bunescu *(Computer Science, University of North Carolina at Charlotte)*,
  Justin Smith *(Computer Science, University of North Carolina at Charlotte)*,
  Ryan Teehan *(Center for Data Science, New York University)*

- **Automating Source Code Refactoring in the Classroom** 60
  Eman Abdullah AlOmar *(Stevens Institute of Technology)*,
  Mohamed Wiem Mkaouer *(University of Michigan)*, Ali Ouni *(ETS Montreal, University of Quebec)*
- **Trust in Generative AI among Students: An exploratory study** 67
  Matin Amoozadeh *(University of Houston)*, David Daniels *(Simon Fraser University)*,
  Daye Nam *(Carnegie Mellon University)*, Aayush Kumar *(Indian Institute of Technology)*,
  Stella Chen *(University of Houston)*, Michael Hilton *(Carnegie Mellon University)*,
  Sruti Srinivasa Ragavan *(Indian Institute of Technology)*, Amin Alipour *(University of Houston)*
- **Foot in the Door: Developing Opportunities for Computing Undergraduates to Gain Industry Experience** 74
  Nimmi Arunachalam *(Florida International University)*, Stephanie J. Lunn *(Florida International University)*,
  Mark Weiss *(Florida International University)*, Jason Liu *(Florida International University)*,
  Giri Narasimhan *(Florida International University)*
- **The Development of Students' Professional Competencies on a Work-Based Software Engineering Program** 81
  Matthew Barr *(University of Glasgow)*, Oana Andrei *(University of Glasgow)*,
  Alistair Morrison *(University of Glasgow)*, Syed Waqar Nabi *(University of Glasgow)*
- **"It Can Relate to Real Lives": Attitudes and Expectations in Justice-centered Data Structures & Algorithms for Non-Majors** 88
  Anna Batra *(University of Washington)*, Iris Zhou *(University of Washington)*,
  Suh Young Choi *(University of Washington)*, Chongjiu Gao *(University of Washington)*,
  Yanbing Xiao *(University of Washington)*, Sonia Fereidooni *(University of Washington)*,
  Kevin Lin *(University of Washington)*
- **Investigating Middle School Students' Early Experience in Learning Computer Science Through Creating Apps for Social Good** 95
  Gillian Bausch *(University at Albany, SUNY)*, Lijun Ni *(University at Albany, SUNY)*,
  Elizabeth Thomas-Cappello *(University at Albany, SUNY)*, Fred Martin *(University of Texas at San Antonio)*,
  Bernado Feliciano *(University of Massachusetts Lowell)*,
  Foozieh Mirderikvand *(University of Massachusetts Lowell)*
- **To Be Or Not To Be . . . An Algorithm: The Notion According to Students and Teachers** 102
  Carlo Bellettini *(Department of Computer Science, Università degli Studi di Milano)*,
  Violetta Lonati *(Department of Computer Science, Lab. CINI Informatica e Scuola, Università degli Studi di Milano)*,
  Mattia Monga *(Department of Computer Science, Lab. CINI Informatica e Scuola, Università degli Studi di Milano)*,
  Anna Morpurgo *(Department of Computer Science, Lab. CINI Informatica e Scuola, Università degli Studi di Milano)*
- **Welcoming Students to Undergraduate Computer Science Programs: On-ramps, Rest Areas, and Lane Changes** 109
  Niharika Bhaskar *(University of California San Diego)*, Amari N. Lewis *(University of California San Diego)*,
  Rona Darabi *(University of California San Diego)*, Joana Fang *(University of California Los Angeles)*,
  Jingting Liu *(University of California San Diego)*, Kristen Vaccaro *(University of California San Diego)*,
  Joe Gibbs Politz *(University of California San Diego)*, Mia Minnes *(University of California San Diego)*
- **What Factors Influence Persistence in Project-based Programming Courses at Community Colleges?** 116
  Christopher Bogart *(Carnegie Mellon University)*, Marshall An *(Carnegie Mellon University)*,
  Eric Keylor *(Carnegie Mellon University)*, Pawanjeet Singh *(Carnegie Mellon University)*,
  JaromÍr Savelka *(Carnegie Mellon University)*, Majd Sakr *(Carnegie Mellon University)*
- **Bringing Social Computing to Secondary School Classrooms** 123
  Kianna Bolante *(University of Washington)*, Kevin Chen *(Cleveland STEM High School)*,
  Quan Ze Chen *(University of Washington)*, Amy Zhang *(University of Washington)*
- **The Case for LLM Workshops: The Responsible Use of Large Language Models by Faculty at Small Liberal Arts Universities** 130
  Chris Bopp *(Computer Science, St. Bonaventure University)*,
  Anne Foerst *(Computer Science, St. Bonaventure University)*,
  Brian Kellogg *(Computer Science and Cybersecurity, St. Bonaventure University)*

- **Designing and Delivering a Post-Quantum Cryptography Course** 137
  Thomas J. Borrelli *(Department of Computer Science, Rochester Institute of Technology)*,
  Monika Polak *(Department of Computer Science, University of Rochester)*,
  Stanisław Radziszowski *(Department of Computer Science, Rochester Institute of Technology)*
- **GitKit: Learning Free and Open Source Collaboration in Context** 144
  Grant Braught *(Dickinson College)*, Stoney Jackson *(Western New England University)*,
  Cam Macdonell *(MacEwan University)*, Lori Postner *(Nassau Community College)*,
  Wesley Shumar *(Drexel University)*, Karl R. Wurst *(Worcester State University)*
- **Reaching Black Women Interested in Computing: The Importance of Organizational Ties** .. 151
  Bailey Brown *(Spelman College)*, Rebecca Zarch *(SageFox Consulting Group)*,
  Amanda Menier *(SageFox Consulting Group)*, Talia Goldwasser *(SageFox Consulting Group)*,
  Megean Garvin *(Maryland Center for Computing Education at the University System of Maryland)*,
  Celeste Lee *(Spelman College)*, Jayce Warner *(Gibson Consulting Group)*,
  Tamara Pearson *(Georgia Institute of Technology)*
- **Confidence vs Insight: Big and Rich Data in Computing Education Research** 158
  Neil C. C. Brown *(King's College London)*, Mark Guzdial *(University of Michigan)*,
- **Writing Between the Lines: How Novices Construct Java Programs** 165
  Neil C. C. Brown *(King's College London)*, Victoria Mac *(King's College London)*,
  Pierre Weill-Tessier *(King's College London)*, Michael Kölling *(King's College London)*
- **Use of AI-driven Code Generation Models in Teaching and Learning Programming: A Systematic Literature Review** 172
  Doga Cambaz *(EEMCS, Delft University of Technology)*, Xiaoling Zhang *(EEMCS, Delft University of Technology)*
- **Generating Multi-Part Autogradable Faded Parsons Problems From Code-Writing Exercises** 179
  Serena Caraco *(Department of Electrical Engineering and Computer Sciences, University of California, Berkeley)*,
  Nelson Lojo *(Department of Electrical Engineering and Computer Sciences, University of California, Berkeley)*,
  Michael Verdicchio *(Department of Cyber and Computer Sciences, The Citadel)*,
  Armando Fox *(Department of Electrical Engineering and Computer Sciences, University of California, Berkeley)*
- **From Degree to Developer: The Creation and Evolution of a CS Course Designed to Bridge the Academia-Industry Gap** 186
  Jonathan Cazalas *(Florida Southern College)*, Christian Roberson *(Florida Southern College)*,
  Zeeshan Furqan *(Warner Bros. Discovery)*
- **Implementation of Split Deadlines in a Large CS1 Course** 193
  Hongxuan Chen *(University of Illinois at Urbana-Champaign)*, Ang Li *(University of Illinois at Urbana-Champaign)*,
  Geoffrey Challen *(University of Illinois at Urbana-Champaign)*,
  Kathryn Cunningham *(University of Illinois at Urbana-Champaign)*
- **Brief, Just-in-Time Teaching Tips to Support Computer Science Tutors** 200
  Alan Y. Cheng *(Stanford University)*, Ellie Tanimura *(Stanford University)*,
  Joseph Tey *(Stanford University)*, Andrew C. Wu *(Stanford University)*, Emma Brunskill *(Stanford University)*
- **Effect of Deadlines on Student Submission Timelines and Success in a Fully-Online Self-Paced Course** 207
  Meng-Chieh Chiu *(Manning College of Information and Computer Sciences, University of Massachusetts Amherst)*,
  Eliot Moss *(Manning College of Information and Computer Sciences, University of Massachusetts Amherst)*,
  Timothy Richards *(Manning College of Information and Computer Sciences, University of Massachusetts Amherst)*
- **Procrastination vs. Active Delay: How Students Prepare to Code in Introductory Programming** 214
  Elizabeth B. Cloude *(Tampere Univesrity)*, Jiayi Zhang *(University Pennsylvania)*,
  Ryan S. Baker *(University of Pennsylvania)*, Eric Fouh *(University of Pennsylvania)*
- **Small Steps, Big Progress: Analyzing District Led Goals to Advance CS Education** 221
  Alexis Cobo *(CSforALL)*, Stephanie B. Wortel-London *(CSforALL)*,
  Leigh Ann DeLyser *(CSforALL)*, Darius Ellis James *(Code Crew)*
- **Talking Techquity: Teaching the Equity and Social Justice Impacts of Computing in Middle School Classrooms** 227
  Merijke Coenraad *(Learning Experience Design, Digital Promise Global)*,
  David Weintrop *(Teaching & Learning, Policy & Leadership, University of Maryland)*

- **How do Computing Students Conceptualize Cybersecurity? Survey Results and Strategies for Curricular Integration** 234
  Noah Q. Cowit *(Department of Information Science, University of Colorado Boulder)*,
  Vidushi Ojha *(Department of Computer Science, University of Illinois at Urbana-Champaign)*,
  Casey Fiesler *(Department of Information Science, University of Colorado Boulder)*
- **A Critical Review of Cybersecurity Education in the United States** 241
  James Crabb *(School of Electrical Engineering and Computer Science, Washington State University)*,
  Christopher Hundhausen *(School of Electrical Engineering and Computer science, Oregon State University)*,
  Assefaw Gebremedhin *(School of Electrical Engineering and Computer science, Washington State University)*
- **Seeing the Whole Elephant — A Comprehensive Framework for Data Education** 248
  Iain J. Cruickshank *(United States Military Academy)*, Nathaniel D. Bastian *(United States Military Academy)*,
  Jean R.S. Blair *(United States Military Academy)*, Christa M. Chewar *(United States Military Academy)*,
  Edward Sobiesk *(United States Military Academy)*
- **How Pre-class Programming Experience Influences Students' Contribution to Their Team Project: A Statistical Study** 255
  Jialin Cui *(North Carolina State University)*, Runqiu Zhang *(University of Virginia)*,
  Ruochi Li *(North Carolina State University)*, Fangtong Zhou *(North Carolina State University)*,
  Yang Song *(University of North Carolina Wilmington)*, Edward Gehringer *(North Carolina State University)*
- **Gender, Social Interactions and Interests of Characters Illustrated in Scratch and Python Programming Books for Children** 262
  Shirley de Wit *(Software Technology, Delft University of Technology)*,
  Felienne Hermans *(Computer Science, Vrije Universiteit Amsterdam)*,
  Marcus Specht *(Software Technology, Delft University of Technology)*,
  Efthimia Aivaloglou *(Software Technology, Delft University of Technology)*
- **Bite-Sized Experiential Education for Computer and Information Science** 269
  Julia Dean *(University of Colorado Boulder)*, Lecia Barker *(University of Colorado Boulder)*,
  Amy Voida *(University of Colorado Boulder)*
- **Transforming Grading Practices in the Computing Education Community** 276
  Adrienne Decker *(Department of Engineering Education, University at Buffalo)*,
  Stephen H. Edwards *(Computer Science, Virginia Tech)*,
  Brian M. McSkimming *(Engineering Pathways, University of Oklahoma)*,
  Bob Edmison *(Computer Science, Virginia Tech)*, Audrey Rorrer *(Computer Science, UNC Charlotte)*,
  Manuel A. Pérez Quiñones *(Software and Information Systems, UNC Charlotte)*
- **Virtual Peer Mentoring to Develop a Sense of Belonging During COVID-19 – A Pilot Study**. 283
  Sonal Dekhane *(Georgia Gwinnett College)*, Hyesung Park *(Georgia Gwinnett College)*,
  Lorraine Jonassen *(Georgia Gwinnett College)*, Wei Jin *(Georgia Gwinnett College)*
- **Evaluating Automatically Generated Contextualised Programming Exercises** 289
  Andre del Carpio Gutierrez *(University of Auckland)*, Paul Denny *(University of Auckland)*,
  Andrew Luxton-Reilly *(University of Auckland)*
- **Prompt Problems: A New Programming Exercise for the Generative AI Era** 296
  Paul Denny *(University of Auckland)*, Juho Leinonen *(University of Auckland)*,
  James Prather *(Abilene Christian University)*, Andrew Luxton-Reilly *(University of Auckland)*,
  Thezyrie Amarouche *(University of Toronto Scarborough)*,
  Brett A. Becker *(University College Dublin)*, Brent N. Reeves *(Abilene Christian University)*
- **Evaluating Mastery-oriented Grading in an Intensive CS1 Course** 303
  Igor dos Santos Montagner *(Insper)*, Rafael Corsi Ferrão *(Insper)*, Andrew Kurauchi *(Insper)*,
  Mariana Silva *(University of Illinois)*, Craig Zilles *(University of Illinois)*
- **Experiences Using Research Processes in an Undergraduate Theory of Computing Course** .. 310
  Ryan E. Dougherty *(Department of Electrical Engineering and Computer Science, United States Military Academy)*
- **One Solution to Addressing Assessment Logistical Problems: An Experience Setting Up and Operating an In-person Testing Center** 317
  Kelly Downey *(Computer Science and Engineering, University of California, Riverside)*,
  Kris Miller *(Computer Science and Engineering, University of California, Riverside)*,
  Mariana Silva *(Computer Science, University of Illinois at Urbana-Champaign)*,
  Craig Zilles *(Computer Science, University of Illinois at Urbana-Champaign)*

- **Beyond HCI: The Need for Accessibility Across the CS Curriculum** 324
  Yasmine N. Elglaly *(Western Washington University)*, Catherine M. Baker *(Creighton University)*, Anne Spencer Ross *(Bucknell University)*, Kristen Shinohara *(Rochester Institute of Technology)*
- **Rubric for the Quality of Answers to Student Queries about Code** 331
  Svana Esche *(Technical University of Darmstadt)*
- **Towards Establishing a Training Program to Support Future CS Teaching-focused Faculty** .. 338
  Mohammed Farghally *(Department of Computer Science, Virginia Tech)*, Mohammed Seyam *(Department of Computer Science, Virginia Tech)*, Cliff A. Shaffer *(Department of Computer Science, Virginia Tech)*
- **CS1 with a Side of AI: Teaching Software Verification for Secure Code in the Era of Generative AI** 345
  Amanda S. Fernandez *(University of Texas at San Antonio)*, Kimberly A. Cornell *(University at Albany)*
- **A High School Camp on Algorithms and Coding in a Small Island Developing State** 352
  Daniel T. Fokum *(University of the West Indies, Mona)*, Zaria Chen Shui *(University of the West Indies, Mona)*, Kerene Wright *(University of the West Indies, Mona)*, Orr Paradise *(University of California, Berkeley)*, Gunjan Mansingh *(University of the West Indies, Mona)*, Daniel Coore *(University of the West Indies, Mona)*
- **Exploring Computing Students' Sense of Belonging Before and After a Collaborative Learning Course** 359
  Morgan M. Fong *(Department of Computer Science, University of Illinois, Urbana-Champaign)*, Shan Huang *(Department of Computer Science, University of Illinois, Urbana-Champaign)*, Abdussalam Alawini *(Department of Computer Science, University of Illinois, Urbana-Champaign)*, Mariana Silva *(Department of Computer Science, University of Illinois, Urbana-Champaign)*, Geoffrey L. Herman *(Department of Computer Science, University of Illinois, Urbana-Champaign)*
- **Application of Collaborative Learning Paradigms within Software Engineering Education: A Systematic Mapping Study** 366
  Rita Garcia *(Victoria University of Wellington)*, Christoph Treude *(University of Melbourne)*, Andrew Valentine *(University of Melbourne)*
- **Socially Responsible Computing in an Introductory Course** 373
  Aakash Gautam *(University of Pittsburgh)*, Anagha Kulkarni *(San Francisco State University)*, Sarah Hug *(Colorado Evaluation & Research Consulting)*, Jane Lehr *(California Polytechnic State University, San Luis Obispo)*, Ilmi Yoon *(San Francisco State University)*
- **Using Embedded Xinu to Teach Operating Systems on Baremetal RISC-V** 380
  Alexander Gebhard *(Marquette University)*, Jack Forden *(Marquette University)*, Oliver Laufenberg *(Marquette University)*, Dennis Brylow *(Marquette University)*
- **Analyzing-Evaluating-Creating: Assessing Computational Thinking and Problem Solving in Visual Programming Domains** 387
  Ahana Ghosh *(MPI-SWS)*, Liina Malva *(MPI-SWS & Tallinn University)*, Adish Singla *(MPI-SWS)*
- **Remote Controlled Cyber: Toward Engaging and Educating a Diverse Cybersecurity Workforce** 394
  Curtice Gough *(L3Harris Institute for Assured Information, Florida Institute of Technology)*, Carl Mann *(L3Harris Institute for Assured Information, Florida Institute of Technology)*, Cherrise Ficke *(Florida Institute of Technology)*, Maureen Namukasa *(Florida Institute of Technology)*, Meredith Carroll *(Florida Institute of Technology)*, TJ Oconnor *(L3Harris Institute for Assured Information, Florida Institute of Technology)*
- **"Help Me Solve It" or "Solve It For Me": Effects of Feedback on Children Building and Programming Robots** 401
  Luisa Greifenstein *(University of Passau)*, Isabella Graßl *(University of Passau)*, Ute Heuer *(University of Passau)*, Gordon Fraser *(University of Passau)*
- **Hint Cards for Common Ozobot Robot Issues: Supporting Feedback for Learning Programming in Elementary Schools** 408
  Luisa Greifenstein *(Universität Passau)*, Ute Heuer *(Universität Passau)*, Gordon Fraser *(University of Passau)*
- **Experience Report: Meet the Professor - A Large-Course Intervention for Increasing Rapport** 415
  William G. Griswold *(University of California, San Diego)*

- **Teaching AI to K-12 Learners: Lessons, Issues, and Guidance** 422
  Shuchi Grover *(Looking Glass Ventures (Edfinity)*
- **Comparing the Security of Three Proctoring Regimens for Bring-Your-Own-Device Exams** 429
  Rishi Gulati *(University of Illinois at Urbana-Champaign)*,
  Matthew West *(University of Illinois at Urbana-Champaign)*,
  Craig Zilles *(University of Illinois at Urbana-Champaign)*,
  Mariana Silva *(University of Illinois at Urbana-Champaign)*
- **Attribution of Work in Programming Teams with Git Reporter** 436
  Michael Guttmann *(Graz University of Technology)*, Aleksandar Karakaš *(Graz University of Technology)*,
  Denis Helic *(Modul University Vienna GmbH)*
- **Using Worked Examples for Engaging in Epistemic Programming Projects** 443
  Sven Hüsing *(Paderborn University)*, Carsten Schulte *(Paderborn University)*,
  Sören Sparmann *(Paderborn University)*, Mario Bolte *(Paderborn University)*
- **An Experience Report: Integrating Oral Communication and Public Speaking Training in a CS Capstone Course** 450
  Shirin Haji Amin Shirazi *(University of California, Riverside)*,
  Mariam Salloum *(University of California, Riverside)*, Annika Speer *(University of California, Riverside)*,
  Neftali Watkinson *(University of California, Riverside)*
- **Industry Mentoring and Internship Experiences at a Community College Baccalaureate Program in Software Development** 456
  Kendrick Hang *(Green River College)*, Tyler Schrock *(Green River College)*,
  Tina J. Ostrander *(Green River College)*, Roseann Berg *(Green River College)*, Tyler Menezes *(CodeDay)*,
  Kevin Wang *(Mentors in Tech)*
- **Conceptual Metaphor Theory in Action: Insights into Student Understanding of Computing Concepts** 463
  Colton Harper *(University of Nebraska-Lincoln)*, Keith Tran *(North Carolina State University)*,
  Stephen Cooper *(University of Nebraska-Lincoln)*
- **Putting the Service into Service Learning: A Report on a Survey of CS Faculty** 470
  Avery Harrell *(Department of Information Science, University of Colorado Boulder)*,
  Sidney Lentz *(Department of Information Science, University of Colorado Boulder)*,
  Fujiko Robledo Yamamoto *(Department of Information Science, University of Colorado Boulder)*,
  Amy Voida *(Department of Information Science, University of Colorado Boulder)*,
  Lecia Barker *(Department of Information Science, University of Colorado Boulder)*
- **Specifications and Contract Grading in Computer Science Education** 477
  Brian Harrington *(University of Toronto Scarborough)*, Abdalaziz Galal *(University of Toronto)*,
  Rohita Nalluri *(University of Toronto Scarborough)*, Faiza Nasiha *(University of Toronto Scarborough)*,
  Anagha Vadarevu *(University of Toronto Scarborough)*
- **Empowering Digital Natives: InstaClone - A Novel Approach to Data Literacy Education in the Age of Social Media** 484
  Anna Hartl *(Research on Learning and Instruction, Technical University of Munich)*,
  Elena Starke *(Computing Education Research, Technical University of Munich)*,
  Angelina Voggenreiter *(Computational Social Science, Technical University of Munich)*,
  Doris Holzberger *(Research on Learning and Instruction, Technical University of Munich)*,
  Tilman Michaeli *(Computing Education Research, Technical University of Munich)*,
  Jürgen Pfeffer *(Computational Social Science, Technical University of Munich)*
- **Neurodiverse Programmers and the Accessibility of Parsons Problems: An Exploratory Multiple-Case Study** 491
  Carl Haynes-Magyar *(Carnegie Mellon University)*
- **Clearn: A Cost-conscious Student-led Online Judge for a Large Programming Course** 498
  Valeria Herskovic *(Department of Computer Science, Pontificia Universidad Católica de Chile)*,
  Jorge Munoz-Gama *(Pontificia Universidad Católica de Chile)*,
  Fernando Balladares *(Pontificia Universidad Católica de Chile)*,
  Pablo Flores *(Pontificia Universidad Católica de Chile)*, Nicolas Quiroz *(Pontificia Universidad Católica de Chile)*
- **Students Investigating Pedagogy: A Project for Learning about Learning in CS** 505
  Lauren Himbeault *(University of Manitoba)*, Celine Latulipe *(University of Manitoba)*

- **Challenges and Approaches to Teaching CS1 in Prison** 512
  Emma Hogan *(UC San Diego)*, Ruoxuan Li *(UC San Diego)*, Adalbert Gerald Soosai Raj *(UC San Diego)*, William G. Griswold *(UC San Diego)*, Leo Porter *(UC San Diego)*
- **Advancing Automated Assessment Tools — Opportunities for Innovations in Upper-level Computing Courses: A Position Paper** 519
  Steffan Hooper *(University of Auckland)*, Burkhard C. Wünsche *(University of Auckland)*, Andrew Luxton-Reilly *(University of Auckland)*, Paul Denny *(University of Auckland)*, Haoran Feng *(University of Auckland)*
- **Detecting ChatGPT-Generated Code Submissions in a CS1 Course Using Machine Learning Models** 526
  Muntasir Hoq *(North Carolina State University)*, Yang Shi *(North Carolina State University)*, Juho Leinonen *(The University of Auckland)*, Damilola Babalola *(North Carolina State University)*, Collin Lynch *(North Carolina State University)*, Thomas Price *(North Carolina State University)*, Bita Akram *(North Carolina State University)*
- **Do Embedded Ethics Modules Have Impact Beyond the Classroom?** 533
  Diane Horton *(Department of Computer Science, University of Toronto)*, David Liu *(Department of Computer Science, University of Toronto)*, Sheila A. McIlraith *(Department of Computer Science, University of Toronto)*, Steven Coyne *(Department of Philosophy, University of Toronto)*, Nina Wang *(Department of Philosophy, Cornell University)*
- **Making Abstraction Concrete in the Elementary Classroom** 540
  Eping E. Hung *(Computer Science, Southern Oregon University)*, Maggie Vanderberg *(Computer Science, Southern Oregon University)*, Gladys Krause *(School of Education, William & Mary)*, Eva Skuratowicz *(Division of Social Sciences, Southern Oregon University)*
- **Scaffolding Minority High School Students' Computer Science Learning: Culturally Relevant Summer Camp** 547
  Jung Won Hur *(Auburn University)*, Jay N. Bhuyan *(Tuskegee University)*, Fan Wu *(Tuskegee University)*
- **Exploring the Effects of a Collaborative Guided Inquiry Learning Approach on Performance and Retention of Underrepresented Minority Students across Multiple Sections in an Introductory Programming Course** 554
  Paul Salvador Inventado *(California State University Fullerton)*, Joshua Caleb Dy *(Google)*
- **Designing and Piloting a High School CS+X Topics Course** 562
  Kathleen Isenegger *(Department of Computer Science, University of Illinois)*, Max Fowler *(Department of Computer Science, University of Illinois)*, Yael Gertner *(Department of Computer Science, University of Illinois)*, Raya Hegeman-Davis *(School of Computing, University of Wyoming)*, Leonard Pitt *(Department of Computer Science, University of Illinois)*
- **Improvement in Program Repair Methods using Refactoring with GPT Models** 569
  Ryosuke Ishizue *(Waseda University & NTT DATA Group Corp.)*, Kazunori Sakamoto *(WillBooster Inc. & Tokyo Online Unicersity)*, Hironori Washizaki *(Waseda University)*, Yoshiaki Fukazawa *(Waseda University)*
- **Adopting Foundational Data Science Curriculum with Diverse Institutional Contexts** 576
  Vandana P. Janeja *(University of Maryland, Baltimore County)*, Maria Sanchez *(University of Maryland, Baltimore County)*, Yi Xuan Khoo *(University of Maryland, Baltimore County)*, Claudia Von Vacano *(University of California, Berkeley)*, Lujie Karen Chen *(University of Maryland, Baltimore County)*
- **PyodideU: Unlocking Python Entirely in a Browser for CS1** 583
  Thomas Jefferson *(Stanford University)*, Chris Gregg *(Stanford University)*, Chris Piech *(Stanford University)*
- **Artificial Intelligence in Compulsory K-12 Computer Science Classrooms: A Scalable Professional Development Offer for Computer Science Teachers** 590
  Franz Jetzinger *(Computing Education Research Group, Technical University of Munich)*, Sven Baumer *(Computing Education Research Group, Technical University of Munich)*, Tilman Michaeli *(Computing Education Research Group, Technical University of Munich)*

- **Experiences of Undergraduate Computer Science Students Living with Mental Health Conditions** 597
  Jie Ji *(Bryn Mawr College)*, Christian Murphy *(Swarthmore College)*, Brianna Blaser *(University of Washington)*, Jennifer Akullian *(Growth Coaching Institute)*
- **Understanding California's Computer Science Transfer Pathways** 604
  Jinya Jiang *(UC San Diego)*, Richa Kafle *(UC San Diego)*, Christa Lehr *(Southern Connecticut State Univ.)*, Simone Wright *(UC San Diego)*, Clarissa Guitierrez-Godoy *(UC San Diego)*, Christine Alvarado *(UC San Diego)*
- **Smart Use of Smart Devices in Your Home: A Smart Home Security and Privacy Workshop for the General Public** 611
  Tushar M. Jois *(City College of New York)*, Tina Pavlovich *(Dartmouth College)*, Brigid M. McCarron *(Dartmouth College)*, David Kotz *(Dartmouth College)*, Timothy J. Pierson *(Dartmouth College)*
- **Need a Programming Exercise Generated in Your Native Language? ChatGPT's Got Your Back: Automatic Generation of Non-English Programming Exercises Using OpenAI GPT-3.5** 618
  Mollie Jordan *(North Carolina State University)*, Kevin Ly *(University of California, San Diego)*, Adalbert Gerald Soosai Raj *(University of California, San Diego)*
- **ChatGPT in the Classroom: An Analysis of Its Strengths and Weaknesses for Solving Undergraduate Computer Science Questions** 625
  Ishika Joshi *(Indraprastha Institute of Information Technology, Delhi)*, Ritvik Budhiraja *(Indraprastha Institute of Information Technology, Delhi)*, Harshal Dev *(Indraprastha Institute of Information Technology, Delhi)*, Jahnvi Kadia *(Indraprastha Institute of Information Technology, Delhi)*, Mohammad Osama Ataullah *(Indraprastha Institute of Information Technology, Delhi)*, Sayan Mitra *(Indraprastha Institute of Information Technology, Delhi)*, Harshal D. Akolekar *(Indian Institute of Technology, Jodhpur)*, Dhruv Kumar *(Indraprastha Institute of Information Technology, Delhi)*
- **Discourse Practices in Computer Science Education** 632
  Yvonne Kao *(WestEd)*, David McKinney *(WestEd)*, Sam Berg *(Oakland Unified School District)*, Brenda Tuohy *(Oakland Unified School District)*, Courtney Ortega *(Oakland Unified School District)*
- **Integrating Natural Language Processing in Middle School Science Classrooms: An Experience Report** 639
  Gloria Ashiya Katuka *(University of Florida)*, Srijita Chakraburty *(Indiana University)*, Hyejeong Lee *(Indiana University)*, Sunny Dhama *(University of Florida)*, Toni Earle-Randell *(University of Florida)*, Mehmet Celepkolu *(University of Florida)*, Kristy Elizabeth Boyer *(University of Florida)*, Krista Glazewski *(North Carolina State University)*, Cindy Hmelo-Silver *(Indiana University)*, Tom McKlin *(The Findings Group)*
- **Community Action Computing: A Data-centric CS0 Course** 646
  Ayaan M. Kazerouni *(California Polytechnic State University)*, Jane Lehr *(California Polytechnic State University)*, Zoë Wood *(California Polytechnic State University)*
- **Stump-the-Teacher: Using Student-generated Examples during Explicit Debugging Instruction** 653
  Chris Kerslake *(Simon Fraser University)*
- **Student Perspectives on Assignment Deadline Policies in Computer Science Courses** 659
  Joseph Kim *(Haverford College)*, Christian Murphy *(Swarthmore College)*
- **Software Engineering Education Must Adapt and Evolve for an LLM Environment** 666
  Vassilka D. Kirova *(Nokia Bell Labs)*, Cyril S. Ku *(William Paterson University)*, Joseph R. Laracy *(Seton Hall University)*, Thomas J. Marlowe *(Seton Hall University)*
- **Cultural-Centric Computational Embroidery** 673
  F. Megumi Kivuva *(University of Washington)*, Jayne Everson *(University of Washington)*, Camilo Montes De Haro *(Bellevue College)*, Amy J. Ko *(University of Washington)*
- **Motivated by Inclusion: Evaluating the Impact of Games on Students' Motivation to Design Accessibly Across a Spectrum of Disabilities** 680
  Devorah Kletenik *(Brooklyn College, City University of New York)*, Rachel F. Adler *(University of Illinois Urbana-Champaign)*

- **From Awareness to Action: Teaching Software Accessibility for Neurodiverse Users** 687
  Devorah Kletenik *(Brooklyn College, City University of New York)*, Rachel Minkowitz *(Cornell Tech)*,
  Aleksandra Peric *(Northeastern Illinois University)*,
  Mehmet Sahin *(Brooklyn College, City University of New York)*,
  Rachel F. Adler *(University of Illinois Urbana-Champaign)*
- **Micro-internships and Career Focused Programs as Mechanisms for Diversifying Computing** 694
  Kristina Kramarczuk *(Department of Teaching & Learning, Policy & Leadership, University of Maryland)*,
  Kate Atchison *(University of Maryland)*, Monica Hilliard *(University of Maryland)*,
  Jandelyn Plane *(Ripon College)*, Sally Bond *(The Program Evaluation Group-LLC)*,
  Caitlin Rudy *(University of Maryland)*, David Weintrop *(Department of Teaching & Learning, Policy & Leadership, University of Maryland)*
- **Worker-Driven Computing Education: A Proof-of-Concept Study for K12 Culturally Responsive Computing** 701
  Michael Lachney *(College of Education, Michigan State University)*,
  Brian Ferguson Bey *(Henry Ford High School, Detroit Public Schools)*,
  Samuela Mouzaoir *(College of Education, Michigan State University)*,
  Christa Robinson *(College of Education, Michigan State University)*
- **Exploring Student Motivation in Integration of Soft Skills Training within Three Levels of Computer Science Programs** 708
  En-Shiun Annie Lee *(Ontario Tech University, University of Toronto)*, Luki Danukarjanto *(FOCUS.inspired)*,
  Sadia Sharmin *(University of Toronto)*, Shou-Yi Hung *(University of Toronto)*,
  Sicong Huang *(University of Toronto)*, Tong Su *(University of Toronto)*
- **Sealing the Deal: Factors that Promote Computing Interns' Interest in Computing Careers** .. 715
  Kathleen J. Lehman *(Education, UCLA)*, Kaitlyn N. Stormes *(Education, UCLA)*,
  Katie N. Smith *(Policy, Organizational & Leadership Studies, Temple University)*,
  Julia C. Lapan *(School of Engineering & Applied Science, University of Virginia)*
- **CONTENTR: An Experiential Game for Teaching Value Tradeoffs in Social Media Governance** 722
  Anna Lenhart *(Univeristy of Maryland)*, Sarah Gilbert *(Cornell University)*,
  Katie Shilton *(Univeristy of Maryland)*
- **Does Curricular Complexity in Computer Science Influence the Representation of Women CS Graduates?** 729
  Albert Lionelle *(Khoury College of Computer Sciences, Northeastern University)*,
  McKenna Quam *(Khoury College of Computer Sciences, Northeastern University)*,
  Carla Brodley *(Center for Inclusive Computing, Northeastern University)*,
  Catherine Gill *(Center for Inclusive Computing, Northeastern University)*
- **A Fast and Accurate Machine Learning Autograder for the Breakout Assignment** 736
  Evan Zheran Liu *(Stanford University & Imbue)*, David Yuan *(Stanford University)*,
  Ahmed Ahmed *(Stanford University)*, Elyse Cornwall *(Stanford University)*,
  Juliette Woodrow *(Stanford University)*, Kaylee Burns *(Stanford University)*,
  Allen Nie *(Stanford University)*, Emma Brunskill *(Stanford University)*, Chris Piech *(Stanford University)*,
  Chelsea Finn *(Stanford University)*
- **Beyond Traditional Teaching: Designing a Virtual Teaching Assistant Using LLMs for CS Education** 743
  Mengqi Liu *(Department of Computer Science, McGill University)*,
  Faten M'Hiri *(Department of Computer Science, McGill University)*
- **Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education** 750
  Rongxin Liu *(Harvard University)*, Carter Zenke *(Harvard University)*, Charlie Liu *(Yale University)*,
  Andrew Holmes *(Harvard University)*, Patrick Thornton *(Harvard University)*,
  David J. Malan *(Harvard University)*
- **"I Didn't Know": Examining Student Understanding of Academic Dishonesty in Computer Science** 757
  Michael Liut *(University of Toronto Mississauga)*, Anna Ly *(University of Toronto Mississauga)*,
  Jessica Jia-Ni Xu *(University of Toronto)*, Justice Banson *(Western Washington University)*,
  Paul Vrbik *(The University of Queensland)*, Caroline D. Hardin *(Western Washington University)*

- **Novel Insights into Elementary Girls' Experiences in Physiological Computing** 764
  Feiya Luo *(The University of Alabama)*, Ruohan Liu *(University of Virginia)*,
  Idowu David Awoyemi *(The University of Alabama)*,
  Chris Crawford *(The University of Alabama)*, Fatema Nasrin *(The University of Alabama)*
- **Breaking Stereotypes and Feeding the STEM Pipeline** 771
  Naja A. Mack *(Morgan State University)*, Michael B. Adeleke *(Morgan State University)*,
  Elijah Ballou *(Morgan State University)*, Destiny Davis *(Morgan State University)*,
  Vincent Ingram *(Morgan State University)*, Katlyn Cox *(Morgan State University)*
- **The Integration of Computational Thinking and Making in the Classroom** 778
  David Magda *(University of Florida)*, Christina Gardner-McCune *(University of Florida)*,
  Yerika Jimenez *(University of Florida)*, Sharon Chu *(University of Florida)*,
  Abhishek Kulkarni *(University of Florida)*
- **Learners Teaching Novices: An Uplifting Alternative Assessment** 785
  Ali Malik *(Stanford University)*, Juliette Woodrow *(Stanford University)*, Chris Piech *(Stanford University)*
- **ChemAIstry: A Novel Software Tool for Teaching Model Training in K-8 Education** 792
  Fred Martin *(University of Texas at San Antonio)*, Vaishali Mahipal *(University of Massachusetts Lowell)*,
  Garima Jain *(University of Massachusetts Lowell)*, Srija Ghosh *(University of Massachusetts Lowell)*,
  Ismaila Temitayo Sanusi *(University of Eastern Finland)*
- **A Global Survey of Introductory Programming Courses** 799
  Raina Mason *(Southern Cross University)*, Simon *(Unaffiliated)*, Brett A. Becker *(University College Dublin)*,
  Tom Crick *(Swansea University)*, James H. Davenport *(University of Bath)*
- **The Correctness of the Mental Model of Arrays After Instruction for CS1 Students** 806
  Syeda Fatema Mazumder *(Software and Information Systems, University of North Carolina at Charlotte)*,
  Manuel A. Pérez Quiñones *(Software and Information Systems, University of North Carolina at Charlotte)*
- **Analyzing Differences in Student Engagement Between a Single Narrative Game Intervention and Multiple Narrative Games Intervention in an Undergraduate Computer Organization and Architecture Course** 812
  Declan Andrew McClintock *(Michigan State University)*,
  Charles B. Owen *(Michigan State University)*
- **Piloting a Diagnostic Tool to Measure AP CS Principles Teachers' Knowledge Against CSTA Teacher Standard 1** 819
  Monica M. McGill *(Institute for Advancing)*, Joseph C. Tise *(Institute for Advancing)*,
  Adrienne Decker *(Computing Education)*
- **Iterative Design of a Socially-Relevant and Engaging Middle School Data Science Unit** 826
  David McKinney *(WestEd)*, Chloe Morton *(WestEd)*, Brenda Tuohy *(Oakland Unified School District)*,
  Sam Berg *(Oakland Unified School District)*, Audrey Karlstad *(Oakland Unified School District)*,
  Courtney Ortega *(Oakland Unified School District)*, Zelda Allison *(Oakland Unified School District)*,
  Griffin Munzel *(Oakland Unified School District)*, Max Washburn *(Oakland Unified School District)*,
  Yvonne Kao *(WestEd)*
- **A History of BPC: Lessons from our Past Informing our Future Directions** 833
  Madison Melton *(University of North Carolina at Charlotte)*,
  Audrey Rorrer *(University of North Carolina at Charlotte)*
- **Taxonomy-Based Human Error Assessment for Senior Software Engineering Students** 840
  Benjamin S. Meyers *(Rochester Institute of Technology)*, Andrew Meneely *(Rochester Institute of Technology)*
- **Cyber Funfair: Creating Immersive and Educational Experiences for Teaching Cyber Physical Systems Security** 847
  Alan Mills *(University of the West of England)*, Jonathan White *(University of the West of England)*,
  Phil Legg *(University of the West of England)*
- **ALAN: Assessment-as-Learning Authentic Tasks for Networking** 853
  Sepehr Minagar *(Faculty of Information Technology, Monash University)*,
  Amin Sakzad *(Faculty of Information Technology, Monash University)*,
  Guido Tack *(Faculty of Information Technology, Monash University)*,
  Carsten Rudolph *(Faculty of Information Technology, Monash University)*,
  Judithe Sheard *(Faculty of Information Technology, Monash University)*

- **EIT: Earnest Insight Toolkit for Evaluating Students' Earnestness in Interactive Lecture Participation Exercises** 860
  Mihran Miroyan *(University of California, Berkeley)*, Shiny Weng *(University of California, Berkeley)*, Rahul Shah *(University of California, Berkeley)*, Lisa Yan *(University of California, Berkeley)*, Narges Norouzi *(University of California, Berkeley)*
- **Teaching Formal Languages through Programmed Instruction** 867
  Mostafa Mohammed *(The State University of New York at Buffalo & Assiut University)*, Clifford A. Shaffer *(Virginia Tech)*
- **Failure Artifact Scenarios to Understand High School Students' Growth in Troubleshooting Physical Computing Projects** 874
  Luis Morales-Navarro *(University of Pennsylvania)*, Deborah A. Fields *(Utah State University)*, Deepali Barapatre *(University of Pennsylvania)*, Yasmin B. Kafai *(University of Pennsylvania)*
- **Not Just Training, Also Testing: High School Youths' Perspective-Taking through Peer Testing Machine Learning-Powered Applications** 881
  Luis Morales-Navarro *(University of Pennsylvania)*, Meghan Shah *(University of Pennsylvania)*, Yasmin B. Kafai *(University of Pennsylvania)*
- **The Need for More Justice-Oriented Courses in Undergraduate Computer Science Curricula** 888
  Sukanya Kannan Moudgalya *(University of Tennessee, Knoxville)*, Amanda Zeller *(University of Tennessee, Knoxville)*
- **Fostering Race-Conscious Literacies in Computer Science Teacher Education** 895
  Sukanya Kannan Moudgalya *(University of Tennessee, Knoxville)*
- **Bringing Teachers and Researchers Together through Participatory Design and Cooperative Prototyping in Computing Education** 902
  Line Have Musaeus *(Center for Computational Thinking and Design, Aarhus University)*, Marianne Graves Petersen *(Center for Computational Thinking and Design, Aarhus University)*, Clemens Nylandsted Klokmose *(Center for Computational Thinking and Design, Aarhus University)*
- **Collecting, Analyzing, and Acting on Intersectional, Longitudinal Data and Pass/Fail/Withdraw Rates in Computing Courses** 909
  Felix Muzny *(Khoury College of Computer Sciences, Center for Inclusive Computing, Northeastern University)*, Megan Giordano *(Center for Inclusive Computing, Northeastern University)*, Emma Sommers *(Northeastern University)*, Carla E. Brodley *(Center for Inclusive Computing, Northeastern University)*
- **Cybersecurity as a Crosscutting Concept Across an Undergrad Computer Science Curriculum: An Experience Report** 916
  Azqa Nadeem *(Delft University of Technology & University of Twente)*
- **Experiences Teaching a Wireless for the Internet of Things Course Co-operatively at Multiple Universities** 923
  Nabeel Nasir *(University of Virginia)*, Viswajith Govinda Rajan *(University of Virginia)*, Pat Pannuto *(University of California, San Diego)*, Branden Ghena *(Northwestern University)*, Bradford Campbell *(University of Virginia)*
- **DOJO: Applied Cybersecurity Education in the Browser** 930
  Connor Nelson *(Arizona State University)*, Yan Shoshitaishvili *(Arizona State University)*
- **PWN The Learning Curve: Education-First CTF Challenges** 937
  Connor Nelson *(d, Arizona State University)*, Yan Shoshitaishvili *(Arizona State University)*
- **Crafting Disability Fairness Learning in Data Science: A Student-Centric Pedagogical Approach** 944
  Pax Newman *(Western Washington University)*, Tyanin Opdahl *(Western Washington University)*, Yudong Liu *(Western Washington University)*, Scott Wehrwein *(Western Washington University)*, Yasmine N. Elglaly *(Western Washington University)*

- **Exploring the Impact of Generative AI for StandUp Report Recommendations in Software Capstone Project Development** 951
  Andres Neyem *(Computer Science Department, Pontificia Universidad Catolica de Chile)*,
  Juan Pablo Sandoval *(Computer Science Department, Pontificia Universidad Catolica de Chile)*,
  Marcelo Mendoza *(Computer Science Department, Pontificia Universidad Catolica de Chile)*,
  Leonardo Centellas-Claros *(Computer Science Department, Pontificia Universidad Catolica de Chile)*,
  Luis A. Gonzalez *(Computer Science Department, Pontificia Universidad Catolica de Chile)*,
  Carlos Paredes-Robles *(Computer Science Department, Pontificia Universidad Catolica de Chile)*
- **Using GPT-4 to Provide Tiered, Formative Code Feedback** 958
  Ha Nguyen *(Utah State University)*, Vicki Allan *(Utah State University)*
- **The First Five Years of a Dual Track Programming Series: A Retrospective Analysis** 965
  Christopher Nitta *(Department of Computer Science, University of California, Davis)*,
  Kurt Eiselt *(Department of Computer Science, University of California, Davis)*
- **CATCHing CS Equity: Counselors, Administrators, and Teachers Collaborating Holistically for Systemic Change** 972
  Manee Ngozi M. Nnamani *(MIT Teaching Systems Lab)*, Salome Otero *(MIT Teaching Systems Lab)*,
  Julie M. Smith *(MIT Teaching Systems Lab)*, Josh Sheldon *(MIT Teaching Systems Lab)*,
  Deborah Boisvert *(CSforMA)*, Justin Reich *(MIT Teaching Systems Lab)*
- **Growth in Knowledge of Programming Patterns: A Comparison Study of CS1 vs. CS2 Students** 979
  Sara Nurollahian *(University of Utah)*, Anna N. Rafferty *(Carleton College)*,
  Noelle Brown *(University of Utah)*, Eliane Wiese *(University of Utah)*
- **PWN Lessons Made Easy with Docker: Toward an Undergraduate Vulnerability Research Cybersecurity Class** 986
  TJ Oconnor *(Florida Institute of Technology)*, Alex Schmith *(Florida Institute of Technology)*,
  Chris Stricklan *(Florida Institute of Technology)*, Marco Carvalho *(Florida Institute of Technology)*,
  Sneha Sudhakaran *(Florida Institute of Technology)*
- **Computing Self-Efficacy in Undergraduate Students: A Multi-Institutional and Intersectional Analysis** 993
  Vidushi Ojha *(University of Illinois at Urbana-Champaign)*, Leah West *(Harvey Mudd College)*,
  Colleen M. Lewis *(University of Illinois at Urbana-Champaign)*
- **Undergraduate Student Attitudes towards a Social Justice Context in a Programming Project** 1000
  Aadarsh Padiyath *(University of Michigan)*, Kyle Ashburn *(University of North Carolina)*,
  Barbara Ericson *(University of Michigan)*
- **Analogies and Active Engagement: Introducing Computer Science** 1007
  Jennifer Parham-Mocello *(Oregon State University)*, Martin Erwig *(Oregon State University)*,
  Margaret Niess *(Oregon State University)*
- **Programming-Integrated Mathematics Learning for Future Elementary Teachers and Non-STEM Majors** 1014
  Hyejin Park *(Drake University)*, Eric D. Manley *(Drake University)*
- **Intersectional Biases Within an Introductory Computing Assessment** 1021
  Miranda C. Parker *(San Diego State University)*, He Ren *(University of Washington)*,
  Min Li *(University of Washington)*, Chun Wang *(University of Washington)*
- **Institutional Perspectives on Formal Work-based Learning Programs in the UK** 1028
  Jack Parkinson *(University of Glasgow)*, Sebastian Dziallas *(University of the Pacific)*
- **From Workshops to Classrooms: Faculty Experiences with Implementing Inclusive Design Principles** 1035
  Pankati Patel *(Department of Computer Science and Technology, Kean University)*,
  Dahana Moz-Ruiz *(Department of Computer Science and Technology, Kean University)*,
  Rosalinda Garcia *(School of Electrical Engineering and Computer Science, Oregon State University)*,
  Amreeta Chatterjee *(School of Electrical Engineering and Computer Science, Oregon State University)*,
  Patricia Morreale *(Department of Computer Science and Technology, Kean University)*,
  Margaret Burnett *(School of Electrical Engineering and Computer Science, Oregon State University)*

- **Unpacking the Unique Role of Black Women Computer Science Educators** 1042
  Tamara Pearson *(Georgia Institute of Technology)*, Pamela Leggett-Robinson *(PLR Consulting)*
- **The Diversity-Hire Narrative in CS: Sources, Impacts, and Responses** 1049
  Christopher Perdriau *(Computer Science, University of Illinois at Urbana Champaign)*,
  Vidushi Ojha *(Computer Science, University of Illinois at Urbana Champaign)*,
  Kaitlynn Gray *(Computer Science, Harvey Mudd College)*,
  Brent Lagesse *(Computing & Software Systems, University of Washington Bothell)*,
  Colleen M. Lewis *(Computer Science, University of Illinois at Urbana Champaign)*
- **Disentangling the Learning Gains from Reading a Book Chapter and Completing Proof Blocks Problems** 1056
  Seth Poulsen *(Utah State University)*, Yael Gertner *(University of Illinois Urbana-Champaign)*,
  Hongxuan Chen *(University of Illinois Urbana-Champaign)*,
  Benjamin Cosman *(University of California San Diego)*,
  Matthew West *(University of Illinois at Urbana-Champaign)*,
  Geoffrey L Herman *(University of Illinois at Urbana-Champaign)*
- **Solving Proof Block Problems Using Large Language Models** 1063
  Seth Poulsen *(Utah State University)*, Sami Sarsa *(Aalto University)*,
  James Prather *(Abilene Christian University)*, Juho Leinonen *(University of Auckland)*,
  Brett A. Becker *(University College Dublin)*, Arto Hellas *(Aalto University)*,
  Paul Denny *(University of Auckland)*, Brent N. Reeves *(Abilene Christian University)*
- **A Self-Regulated Learning Framework using Generative AI and its Application in CS Educational Intervention Design** 1070
  Prajish Prasad *(School of Computing and Data Sciences, FLAME University)*,
  Aamod Sane *(School of Computing and Data Sciences, FLAME University)*
- **Scratch-NB: A Scratch Extension for Introducing K-12 Learners to Supervised Machine Learning** 1077
  Patricio Quiroz *(Department of Computer Science, University of Chile)*,
  Francisco J. Gutierrez *(Department of Computer Science, University of Chile)*
- **Playing with Matches: Adopting Gale--Shapley for Managing Student Enrollments Beyond CS2** 1084
  Anna N. Rafferty *(Carleton College)*, David Liben-Nowell *(Carleton College)*,
  Dave R. Musicant *(Carleton College)*, Emy Farley *(Bowdoin College)*,
  Allie Lyman *(Carleton College)*, Ann May *(Carleton College)*
- **Participatory Governance in the Computer Science Theory Classroom** 1091
  Tim Randolph *(Columbia University)*
- **The Choice is Yours: Intersectional Studies versus Studies of Intersectional Populations in Computing Education Research** 1098
  Yolanda A. Rankin *(Emory University)*, Sheena Erete *(University of Maryland College Park)*,
  Jakita O. Thomas *(Auburn University)*, Nichole Pinkard *(Northwestern University)*
- **Visual CryptoED: A Role-Playing and Visualization Tool for K-12 Cryptography Education** 1105
  Pranathi Rayavaram *(University of Massachusetts Lowell)*,
  Sahithi Charitha Dindukuri *(University of Massachusetts Lowell)*,
  Krishna Vellamchety *(Univeristy of Massachusetts Lowell)*, Justin Marward *(University of Massachusetts Lowell)*,
  Maryam Abbasalizadeh *(University of Massachusetts Lowell)*,
  Claire Seungeun Lee *(University of Massachusetts Lowell)*, Sashank Narain *(University of Massachusetts Lowell)*
- **Traditional vs. Flexible Modalities in a Data Structures Class** 1112
  Shanon Reckinger *(University of Illinois Chicago)*, Joe Hummel *(Northwestern University)*,
  Sarah Heckman *(North Carolina State University)*
- **A Longitudinal Study of the Relationship Between Early Undergraduate Research and Academic Outcomes in Computer Science** 1119
  Kamen Redfield *(University of California, San Diego)*, Sukham Sidhu *(University of California, San Diego)*,
  Zackary Glazewski *(University of California, Santa Barbara)*, Cynthia Lee *(Stanford University)*,
  Diba Mirza *(University of California, Santa Barbara)*, Christine Alvarado *(University of California, San Diego)*

- **Developing Culturally Sustaining Elementary Computer Science Education with Indigenous Communities: Lessons Learned through a Research-Practice Partnership** 1126
  Kathryn M. Rich *(American Institutes for Research)*, Marissa Spang *(American Institutes for Research)*, Jill Bowdon *(American Institutes for Research)*, Joseph P. Wilson *(American Institutes for Research)*, Heather Cunningham *(Boot Up Professional Development)*, Mckay Perkins *(Boot Up Professional Development)*
- **Observations on the Design of Program Planning Notations for Students** 1133
  Elijah Rivera *(Brown University)*, Kathi Fisler *(Brown University)*, Shriram Krishnamurthi *(Brown University)*
- **Teaching Ethics & Activism in a Human-Computer Interaction Professional Master's Program** 1140
  Veronica A. Rivera *(Stanford University)*, Norman Makoto Su *(University of California, Santa Cruz)*
- **Attitudes Towards the Use (and Misuse) of ChatGPT: A Preliminary Study** 1147
  Michael P. Rogers *(University of Wisconsin Oshkosh)*, Hannah Miller Hillberg *(University of Wisconsin Oshkosh)*, Christopher L. Groves *(University of Wisconsin Oshkosh)*
- **The Experience of Near-Peer Computing Mentors: Strengthening and Expanding Women's Computing Identities in Undergraduate Interdisciplinary Contexts** 1154
  Jennifer Rosales *(Barnard College)*, Elizabeth Melville *(Columbia University)*, Melissa Wright *(Barnard College)*, Saima Akhtar *(Barnard College)*, Rebecca N. Wright *(Barnard College)*
- **Diverging Assessments: What, Why, and Experiences** 1161
  Amin Sakzad *(Monash University)*, David Paul *(University of New England)*, Judithe Sheard *(Monash University)*, Ljiljana Brankovic *(University of New England)*, Matthew P. Skerritt *(RMIT University)*, Nan Li *(University of Wollongong)*, Sepehr Minagar *(Monash University)*, Simon *(Unaffiliated)*, William Billingsley *(University of New England)*
- **Applying CS0/CS1 Student Success Factors and Outcomes to Biggs' 3P Educational Model** 1168
  Adrian Salguero *(University of California San Diego)*, Ismael Villegas Molina *(University of California San Diego)*, Lauren Elizabeth Margulieux *(Georgia State University)*, Quintin Cutts *(University of Glasgow)*, Leo Porter *(University of California San Diego)*
- **Learning with Style: Improving Student Code-Style Through Better Automated Feedback** 1175
  Liam Saliba *(The University of Melbourne)*, Elisa Shioji *(The University of Melbourne)*, Eduardo Oliveira *(The University of Melbourne)*, Shaanan Cohney *(The University of Melbourne)*, Jianzhong Qi *(The University of Melbourne)*
- **A Quantitative Methodological Review of Research on Broadening Participation in Computing, 2005-2022** 1182
  Linda J. Sax *(Momentum, UCLA School of Education & Information Studies)*, Chantra Nhien *(Momentum, UCLA School of Education & Information Studies)*, Kaitlyn N. Stormes *(Momentum, UCLA School of Education & Information Studies)*
- **Mining Jewels Together: Debating about Programming Threshold Concepts in Large Classes** 1189
  Manuel Selva *(Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG)*, François Broquedis *(Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG)*
- **Curricular and Pedagogical Considerations in Computer Science Education: The Role of Community Colleges for the Next Decade** 1196
  Christian Servin *(El Paso Community College)*, Elizabeth K. Hawthorne *(Rider University)*, Lori Postner *(Nassau Community College)*, Cara Tang *(Portland Community College)*, Cindy S. Tucker *(Bluegrass Community and Technical College)*
- **A Review of Cognitive Apprenticeship Methods in Computing Education Research** 1202
  Anshul Shah *(University of California, San Diego)*, Adalbert Gerald Soosai Raj *(University of California, San Diego)*
- **Working with Large Code Bases: A Cognitive Apprenticeship Approach to Teaching Software Engineering** 1209
  Anshul Shah *(University of California, San Diego)*, Jerry Yu *(University of California, San Diego)*, Thanh Tong *(University of California, San Diego)*, Adalbert Gerald Soosai Raj *(University of California, San Diego)*

- **ClearMind Workshop: An ACT-based Intervention Tailored for Academic Procrastination among Computing Students** 1216
  Yunyi She *(University of California, San Diego)*, Korena S. Klimczak *(Utah State University)*, Michael E. Levin *(Utah State University)*, Soohyun Nam Liao *(University of California, San Diego)*
- **Instructor Perceptions of AI Code Generation Tools – A Multi-Institutional Interview Study** 1223
  Judy Sheard *(Monash University)*, Paul Denny *(University of Auckland)*, Arto Hellas *(Aalto University)*, Juho Leinonen *(University of Auckland)*, Lauri Malmi *(Aalto University)*, Simon *(Unaffiliated)*
- **Implications of ChatGPT for Data Science Education** 1230
  Yiyin Shen *(University of Wisconsin-Madison)*, Xinyi Ai *(University of California San Diego)*, Adalbert Gerald Soosai Raj *(University of California San Diego)*, Rogers Jeffrey Leo John *(Independent Researcher)*, Meenakshi Syamkumar *(University of Wisconsin-Madison)*
- **Examining Intention to Major in Computer Science: Perceived Potential and Challenges** ... 1237
  Naaz Sibia *(University of Toronto)*, Giang Bui *(University of Toronto)*, Bingcheng Wang *(University of Toronto)*, Yinyue Tan *(University of Toronto)*, Angela Zavaleta Bernuy *(University of Toronto)*, Christina Bauer *(University of Vienna)*, Joseph Jay Williams *(University of Toronto)*, Michael Liut *(University of Toronto Mississauga)*, Andrew Petersen *(University of Toronto Mississauga)*
- **Investigating Students' Usage of Self-regulation of Learning Scaffoldings in a Computer-based Programming Learning Environment** 1244
  Leonardo Silva *(University of Coimbra)*, Anabela Gomes *(Coimbra Polytechnic - ISEC)*, António Mendes *(University of Coimbra)*
- **Exploring Novice Programmers' Testing Behavior: A First Step to Define Coding Struggle** 1251
  Gabriel Silva de Oliveira *(North Carolina State University)*, Zhikai Gao *(North Carolina State University)*, Sarah Heckman *(North Carolina State University)*, Collin Lynch *(North Carolina State University)*
- **Investigating Student Mistakes in Introductory Data Science Programming** 1258
  Anjali Singh *(University of Michigan)*, Anna Fariha *(University of Utah)*, Christopher Brooks *(University of Michigan)*, Gustavo Soares *(Microsoft)*, Austin Z. Henley *(Microsoft)*, Ashish Tiwari *(Microsoft)*, Chethan M *(Microsoft)*, Heeryung Choi *(Massachusetts Institute of Technology)*, Sumit Gulwani *(Microsoft)*
- **Pairing Ungrading with Project-Based Learning in CS1 for Inherently Flexible Course Design** 1265
  Gillian Smith *(Interactive Media & Game Development, Worcester Polytechnic Institute)*
- **Artificial Intelligence Unplugged: Designing Unplugged Activities for a Conversational AI Summer Camp** 1272
  Yukyeong Song *(University of Florida)*, Xiaoyi Tian *(University of Florida)*, Nandika Regatti *(University of Florida)*, Gloria Ashiya Katuka *(University of Florida)*, Kristy Elizabeth Boyer *(University of Florida)*, Maya Israel *(University of Florida)*
- **Computer Science Education - What Can We Learn from Japan?** 1279
  Markus Sprenger *(Dresden University of Technology)*, Thiemo Leonhardt *(Dresden University of Technology)*, Nadine Bergner *(RWTH Aachen University)*, Ryuta Yamamoto *(Shizuoka University)*
- **Accessible to Whom? Bringing Accessibility to Blocks** 1286
  Andreas Stefik *(University of Nevada, Las Vegas)*, Willliam Allee *(University of Nevada, Las Vegas)*, Gabriel Contreras *(University of Nevada, Las Vegas)*, Timothy Kluthe *(University of Nevada, Las Vegas)*, Alex Hoffman *(University of Nevada, Las Vegas)*, Brianna Blaser *(University of Washington)*, Richard Ladner *(University of Washington)*
- **Recognizing Patterns in Productive Failure** 1293
  Phil Steinhorst *(Department of Computer Science, University of Münster)*, Christof Duhme *(Department of Computer Science, University of Münster)*, Xiaoyi Jiang *(Department of Computer Science, University of Münster)*, Jan Vahrenhold *(Department of Computer Science, University of Münster)*
- **An Accessible Blocks Language for Students with and without Visual Impairments** 1300
  Moumita Tabassum *(Dept. of Computer Science, University of Alabama)*, Jeff Gray *(Dept. of Computer Science, University of Alabama)*, Derrick Smith *(Dept. of Curriculum & Instruction, University of Alabama in Huntsville)*

- **Fixing Your Own Smells: Adding a Mistake-Based Familiarisation Step When Teaching Code Refactoring** 1307
  Ivan Tan *(Singapore Management University)*, Christopher M. Poskitt *(Singapore Management University)*
- **dcc --help: Transforming the Role of the Compiler by Generating Context-Aware Error Explanations with Large Language Models** 1314
  Andrew Taylor *(University of New South Wales)*, Alexandra Vassar *(University of New South Wales)*, Jake Renzella *(University of New South Wales)*, Hammond Pearce *(University of New South Wales)*
- **Recursion in Secondary Computer Science Education: A Comparative Study of Visual Programming Approaches** 1321
  Sverrir Thorgeirsson *(ETH Zürich)*, Lennart C. Lais *(ETH Zürich)*, Theo B. Weidmann *(ETH Zürich)*, Zhendong Su *(ETH Zürich)*
- **Comparing Cognitive Load Among Undergraduate Students Programming in Python and the Visual Language Algot** 1328
  Sverrir Thorgeirsson *(ETH Zürich)*, Theo B. Weidmann *(ETH Zürich)*, Karl-Heinz Weidmann *(University of Applied Sciences Vorarlberg)*, Zhendong Su *(ETH Zürich)*
- **HARMONIZING Scratch Encore: Scaffolding K-8 Teachers in Customizing Culturally Responsive Computing Materials** 1335
  Minh Tran *(Department of Computer Science, University of Chicago)*, Heather Killen *(Teaching & Learning, Policy & Leadership, University of Maryland)*, Jen Palmer *(UChicago STEM Education, University of Chicago)*, David Weintrop *(Teaching & Learning, Policy & Leadership, University of Maryland)*, Diana Franklin *(Department of Computer Science, University of Chicago)*
- **Digging Data: Using Archaeology to Teach Responsible Data Practices in a Study Abroad Context** 1342
  Michelle Trim *(Computer Science, University of Massachusetts Amherst)*, Anthony Tuck *(Classics, University of Massachusetts Amherst)*, Matthew Rattigan *(Computer Science, University of Massachusetts Amherst)*, Cole Reilly *(Computer Science, University of Massachusetts Amherst)*
- **A Framework that Explores the Cognitive Load of CS1 Assignments Using Pausing Behavior** 1347
  Joshua Urry *(Computer Science, Utah State University)*, John Edwards *(Computer Science, Utah State University)*
- **Experiences Teaching a CS1 Common Course across 7 Institutions** 1354
  Frank Vahid *(Computer Science and Engineering, University of California, Riverside)*, Ashley Pang *(Computer Science and Engineering, University of California, Riverside)*
- **Towards Comprehensive Metrics for Programming Cheat Detection** 1361
  Frank Vahid *(Computer Science and Engineering, University of California, Riverside)*, Ashley Pang *(Computer Science and Engineering, University of California, Riverside)*, Benjamin Denzler *(Computer Science & Engineering, University of California, Riverside)*
- **CS1 Instructors: Flexibility in Content Approaches is Justified, and Can Enable More Cross-University Cooperation** 1368
  Frank Vahid *(Computer Science & Engineering, University of California, Riverside & zyBooks)*
- **Book Club Model for Engaging with Data Science and Ethics: Using Weapons of Math Destruction** 1374
  Tammy VanDeGrift *(Computer Science, Shiley School of Engineering, University of Portland)*
- **U.S. Latines in Computing: A Review of the Literature** 1381
  Ismael Villegas Molina *(University of California, San Diego)*, Audria Montalvo *(University of California, San Diego)*, Adalbert Gerald Soosai Raj *(University of California, San Diego)*
- **HelpMe: Student Help Seeking using Office Hours and Email** 1388
  Kevin Wang *(The University of British Columbia)*, Ramon Lawrence *(The University of British Columbia)*
- **A Large Scale RCT on Effective Error Messages in CS1** 1395
  Sierra Wang *(Stanford University)*, John Mitchell *(Stanford University)*, Chris Piech *(Stanford University)*

- **Idea Builder: Motivating Idea Generation and Planning through Storyboarding** 1402
  Wengran Wang *(North Carolina State University)*, Ally Limke *(North Carolina State University)*, Mahesh Bobbadi *(North Carolina State University)*, Amy Isvik *(North Carolina State University)*, Veronica Catété *(North Carolina State University)*, Tiffany Barnes *(North Carolina State University)*, Thomas W. Price *(North Carolina State University)*
- **A Measurement Invariance Analysis of the Motivation to Teach Computer Science (MTCS) Scale among Female and Male Educators** 1409
  Zhuoying Wang *(Texas Advanced Computing Center, The University of Texas at Austin)*, Nicole D. Martin *(Texas Advanced Computing Center, The University of Texas at Austin)*, Stephanie N. Baker *(Texas Advanced Computing Center, The University of Texas at Austin)*, Madeline Haynes *(Texas Advanced Computing Center, The University of Texas at Austin)*
- **Improving Software Engineering Teamwork with Structured Feedback** 1414
  Victor Huang Weiqi *(Carnegie Mellon University)*, Kori Krueger *(Morgan State University)*, Taya Cohen *(Carnegie Mellon University)*, Michael Hilton *(Carnegie Mellon University)*
- **Evaluating Identity and Belonging in Computer Science Students: Instrument Adaptation and Analysis** 1421
  Stephanie M. Werner *(Illinois Workforce and Education Research Collaborative, University of Illinois System)*, Ying Chen *(Illinois Workforce and Education Research Collaborative, University of Illinois System)*
- **Experiences in Delivering Online CS Teacher Professional Development** 1428
  Jina Wilde *(University of Texas at San Antonio)*, Emiliano Beltran *(University of Texas at San Antonio)*, Michael J. Zawatski *(University of Texas at San Antonio)*, Amanda S. Fernandez *(University of Texas at San Antonio)*, Priya V. Prasad *(University of Texas at San Antonio)*, Timothy T. Yuen *(University of Texas at San Antonio)*
- **A Cybersecurity Summer Camp for High School Students Using Autonomous R/C Cars** 1435
  Myounggyu Won *(Department of Computer Science, University of Memphis)*, Luke Rivers Carrington *(Department of Computer Science, University of Memphis)*, Douglas Manuel Espinoza II *(Department of Computer Science, University of Memphis)*, Mohd Hasan Ali *(Department of Electrical and, University of Memphis)*, Dipankar Dasgupta *(Department of Computer Science, University of Memphis)*
- **AI Teaches the Art of Elegant Coding: Timely, Fair, and Helpful Style Feedback in a Global Course** 1442
  Juliette Woodrow *(Stanford University)*, Ali Malik *(Stanford University)*, Chris Piech *(Stanford University)*
- **Learning Big Data Systems via Emulation** 1449
  Wensheng Wu *(University of Southern California)*
- **Multi-Pronged Pedagogical Approaches to Broaden Participation in Computing and Increase Students' Computing Persistence: A Robustness Analysis of the STARS Computing Corps' Impact on Students' Intentions to Persist in Computing** 1456
  Lauren Gabrielle Wyatt *(Kent State University)*, Susan Rebecca Fisk *(Kent State University)*, Clarissa Thompson *(Kent State University)*, Jamie Payton *(Temple University)*, Veronica Cateté *(North Carolina State University)*, Audrey Rorrer *(University of North Carolina at Charlotte)*, Tiffany Barnes *(North Carolina State University)*, Tom McKlin *(The Findings Group)*
- **Designing for Equity in Education Via Computational Thinking: A Case Study** 1463
  Ryoko Yamaguchi *(Plus Alpha Research and Consulting)*, Cyntrica Eaton *(Eaton Technologies Consulting Group)*
- **Mechanical TA 2: Peer Grading with TA and Algorithmic Support** 1470
  Hedayat Zarkoob *(University of British Columbia)*, Kevin Leyton-Brown *(University of British Columbia)*
- **Student Interaction with Instructor Emails in Introductory and Upper-Year Computing Courses** 1477
  Angela Zavaleta Bernuy *(University of Toronto)*, Runlong Ye *(University of Toronto)*, Naaz Sibia *(University of Toronto)*, Rohita Nalluri *(University of Toronto)*, Joseph Jay Williams *(University of Toronto)*, Andrew Petersen *(University of Toronto Mississauga)*, Eric Smith *(University of Texas at Austin)*, Bogdan Simion *(University of Toronto Mississauga)*, Michael Liut *(University of Toronto Mississauga)*

- **A Peer Grading Approach for Open-ended Programming Projects Based on Binary System and Swiss System** 1484
  Liang Zhang *(Beihang University)*, Tianyi Chen *(Beihang University)*, Yue Zong *(Beihang University)*, Xiaopeng Gao *(Beihang University)*
- **When Coding Meets Biology: The Tension Between Access and Authenticity in a Contextualized Coding Class** 1491
  Austin L. Zuckerman *(Mathematics & Science Education, UC San Diego & San Diego State University)*, Ashley L. Juavinett *(Neurobiology, UC San Diego)*
- **Student Motivations and Expectations for an Introductory Programming Course in Biology** 1498
  Austin L. Zuckerman *(Math and Science Education, UC San Diego & San Diego State University)*, Lily Vo *(UC San Diego & UC Riverside)*, Ashley L. Juavinett *(Neurobiology, UC San Diego)*

Volume 2

Panels

- **AI in Computing Education from Research to Practice** 1521
  Bita Akram *(North Carolina State University)*, Juho Leinonen *(Aalto University)*, Narges Norouzi *(University of California Berkeley)*, James Prather *(Abilene Christian University)*, Lisa Zhang *(University of Toronto)*
- **Managing Authority When Teaching Computing Ethics** 1523
  Emanuelle Burton *(University of Illinois Chicago)*, Kristel Clayville *(University of Illinois Chicago)*, Stacy A. Doore *(Colby College)*, Michael S. Kirkpatrick *(James Madison University)*, Michael Goldweber *(Denison University)*
- **Interviewing the Teaching Faculty Hiring Process** 1525
  Geoffrey Challen *(University of Illinois)*, Victoria Dean *(Olin College of Engineering)*, Nate Derbinsky *(Northeastern University)*, Matt X. Wang *(University of Washington)*, Jacqueline Smith *(University of Toronto)*
- **Computing in Data Science or Data in Computer Science? Exploring the Relationship between Data Science and Computer Science in K-12 Education** 1527
  Zarek Drozda *(The University of Chicago)*, Justice Toshiba Walker *(The University of Texas at El Paso)*, Kathi Fisler *(Brown University)*, David Weintrop *(University of Maryland)*
- **Broadening Participation in Computing Education: Advancing LGBTQIA+ Voices** 1529
  Wendy M. DuBow *(University of Colorado-Boulder)*, Stephanie Jones *(Northwestern University)*, Stacey Sexton *(SageFox Consulting Group)*, Sri Yash Tadimalla *(University of North Carolina)*
- **Technology We Can't Live Without!** 1531
  Daniel Garcia *(UC Berkeley)*, Delnavaz Dastur *(Stratford San Jose Middle School)*, Kimberly Hermans *(Irvine Unified School District)*, Nery Chapeton-Lamas *(MiraCosta College)*, Ariel Ortiz *(Tecnologico de Monterrey)*
- **Enduring Lessons from 'Computer Science for All' for AI Education in Schools** 1533
  Shuchi Grover *(Looking Glass Ventures)*, Deborah Fields *(Utah State University)*, Yasmin Kafai *(University of Pennsylvania)*, Shana White *(Kapor Center)*, Carla Strickland *(UChicago STEM Education)*
- **Re-making CS Departments for Generation CS** 1535
  Kathleen J. Lehman *(University of California, Los Angeles)*, Carla Brodley *(Northeastern University)*, Mark Guzdial *(University of Michigan)*, Paul Tymann *(Rochester Institute of Technology)*, Aman Yadav *(Michigan State University)*
- **Using Computational Thinking to Demystify Computer Science for Elementary Teachers** . 1537
  Diane Levitt *(Cornell Tech & Cornell University)*, Dylana Garfus-Knowles *(SOCS for All, Ashland School District)*, Wyman Khuu *(Science, Technology & Engineering, KIPP NYC)*, Sara Siddappa *(PS 39 Cypress Hills CS, NYC Public Schools)*

- **An Ethical Imperative: Increasing the Participation of Black Women in Computer Science Education Research** 1539
  Tamara Pearson *(Georgia Institute of Technology)*, Stefanie Marshall *(Michigan State University)*, Yolanda A. Rankin *(Florida State University)*, Mia S. Shaw *(New York University)*
- **A Panel for Distinctive Computing Curricula: Programs Aligned with Liberal Arts Institutional and Departmental Identity** 1541
  Andrea Tartaro *(Furman University)*, Amanda Holland-Minkley *(Washington & Jefferson College)*, Mario Nakazawa *(Berea College)*, David Reed *(Creighton University)*
- **Equitable Professional Learning for K12 Computer Science Teachers** 1543
  Shana V. White *(Dir, CS Equity Kapor Foundation)*, Allison Scott *(CEO, Kapor Foundation)*, Aman Yadav *(College of Education, Michigan State University)*, Diane Levitt *(Dir of K12 Education, Cornell Tech)*

Posters

- **Using High School Student Perspectives to Develop an IoT-based CS Curriculum** 1546
  Pramod Abichandani *(New Jersey Institute of Technology)*, Craig Iaboni *(New Jersey Institute of Technology)*, Prateek Shekhar *(New Jersey Institute of Technology)*
- **Evolution of an Integrated, Elementary CSforAll Curriculum** 1548
  W. Richards Adrion *(Manning College of Information & Computer Sciences, University of Massachusetts)*, Emrah Pektaş *(College of Education, University of Massachusetts)*
- **Understanding the Role of Temperature in Diverse Question Generation by GPT-4** 1550
  Arav Agarwal *(Carnegie Mellon University)*, Karthik Mittal *(Carnegie Mellon University)*, Aidan Doyle *(Carnegie Mellon University)*, Pragnya Sridhar *(Carnegie Mellon University)*, Zipiao Wan *(Carnegie Mellon University)*, Jacob Arthur Doughty *(Carnegie Mellon University)*, Jaromir Savelka *(Carnegie Mellon University)*, Majd Sakr *(Carnegie Mellon University)*
- **An Investigation on Task Difficulty: Does Task Difficulty Depend on the Technology Used in Task Completion?** 1552
  Mahir Akgun *(Pennsylvania State University)*, Sacip Toker *(Atilim University)*
- **Elementary Teachers Engaging with Learning Trajectories to Create Professional Learning Goals around Computer Science Integration** 1554
  Jennifer Albert *(Zucker Family School of Education, The Citadel)*, Candace Joswick *(Dept. of Curriculum and Instruction, University of Texas at Arlington)*, Deepti Joshi *(Dept. of Cyber and Computer Sciences, The Citadel)*, Robin Jocius *(Dept. of Curriculum and Instruction, University of Texas at Arlington)*, Melanie Blanton *(Zucker Family School of Education, The Citadel)*, Robert Petrulis *(EPRE Consulting)*
- **Achieving Diversity in AI-focused Graduate Research Traineeships** 1556
  Cecilia O. Alm *(Rochester Institute of Technology)*, Esa Rantanen *(Rochester Institute of Technology)*, Kristen Shinohara *(Rochester Institute of Technology)*, Ferat Sahin *(Rochester Institute of Technology)*, Chelsea BaileyShea *(Compass Evaluation)*, Reynold Bailey *(Rochester Institute of Technology)*
- **Designing Create Tech: A Discussion of Previous Camp Iterations and Outcomes** 1558
  Charlotte Avery *(Department of Computer Science, University of Maryland)*
- **Considerations for Improving Comprehensive Undergraduate Computing Ethics Education** 1560
  Grace Barkhuff *(Georgia Institute of Technology)*, Jason Borenstein *(Georgia Institute of Technology)*, Daniel Schiff *(Purdue University)*, Judith Uchidiuno *(Georgia Institute of Technology)*, Ellen Zegura *(Georgia Institute of Technology)*
- **Should We Teach Computing Ethics to Master's Students?** 1562
  Grace Barkhuff *(Georgia Institute of Technology)*, Judith Uchidiuno *(Georgia Institute of Technology)*, Ellen Zegura *(Georgia Institute of Technology)*
- **Sokoban: An Assignment for an Object-Oriented and GUI Programming Course** 1564
  João Paulo Barros *(School of Technology and Management, Polytechnic Institute of Beja)*

- **Special Education Teachers Evaluating the Accessibility of CS Educational Robotics** 1566
  Andrew Bennett *(University of Florida)*, Maya Israel *(University of Florida)*,
  Joanne Barrett *(University of Florida)*, Debra "Kelly" Thomas *(Broward County, FL Public Schools)*,
  Jason McKenna *(VEX Robotics)*
- **Supporting Mastery Learning Through an Adaptive Grade Portal** 1568
  Connor Robert Bernard *(University of California, Berkeley)*, Dev Ahluwalia *(University of California, Berkeley)*,
  Madelen Flores *(University of California, Berkeley)*, Wilson Chu *(University of California, Berkeley)*,
  Yuanhan Li *(University of California, Berkeley)*, Daniel Garcia *(University of California, Berkeley)*
- **Co-Designing Integrated CS Curriculum Artifacts with K-5 Classroom Teachers** 1570
  Maverick Berner *(Marquette University)*, Maximus Berner *(Marquette University)*,
  Darcy Ronan *(Sacred Heart University)*, Dennis Brylow *(Marquette University)*
- **Using Notional Machines to Automatically Assess Students' Comprehension of Their Own Code** 1572
  Joey Bevilacqua *(Software Institute, Università della Svizzera italiana)*,
  Luca Chiodini *(Software Institute, Università della Svizzera italiana)*,
  Igor Moreno Santos *(Software Institute, Università della Svizzera italiana)*,
  Matthias Hauswirth *(Software Institute, Università della Svizzera italiana)*
- **My Learnings from Allowing Large Language Models in Introductory Computer Science Classes** 1574
  Rasika Bhalerao *(Northeastern University)*
- **Effects of Prior Academic Experience in Introductory Level Data Science Course** 1576
  Noshaba Bhalli *(University of Maryland, Baltimore County)*,
  Vandana Janeja *(University of Maryland, Baltimore County)*,
  David Harding *(University of California, Berkeley)*
- **Do Hints Enhance Learning in Programming Exercises? Exploring Students' Problem-Solving and Interactions** 1578
  Giang Bui *(University of Toronto Mississauga)*, Nicholas Susanto *(University of Toronto Mississauga)*,
  Naaz Sibia *(University of Toronto)*, Angela Zavaleta Bernuy *(University of Toronto)*,
  Michael Liut *(University of Toronto Mississauga)*, Andrew Petersen *(University of Toronto Mississauga)*
- **A Comparative Analysis of Tools & Task Types for Measuring Computational Problem-Solving** 1580
  Engin Bumbacher *(Haute École Pédagogique du Canton de Vaud (HEP-VD))*,
  Jérôme Brender *(Haute École Pédagogique du Canton de Vaud (HEP-VD))*,
  Richard Lee Davis *(École Polytechnique Fédérale de Lausanne (EPFL))*
- **Integrating & Implementing K-12 Computing Pathways across Six School Districts—Challenges & Opportunities** 1582
  Quinn Burke *(Learning Experience Design, Digital Promise Global)*,
  Merijke Coenraad *(Learning Experience Design, Digital Promise Global)*,
  Alessandra Rangel *(Learning Experience Design, Digital Promise Global)*,
  Kyle Dunbar *(Learning Experience Design, Digital Promise Global)*,
  Kelly Mills *(Learning Experience Design, Digital Promise Global)*
- **Analyzing Student and Instructor Comments using NLP** 1584
  Zack Butler *(Rochester Inst. of Tech.)*, Ivona Bezáková *(Rochester Inst. of Tech.)*,
  Shaoxuan Xu *(Rochester Inst. of Tech.)*, Angelina Brilliantova *(Rochester Inst. of Tech.)*
- **Parallel Islands: A Parallel Computing Educational Video Game** 1586
  Melissa Cameron *(Virginia Tech)*, Margaret Ellis *(Virginia Tech)*, Dimitrios Nikolopoulos *(Virginia Tech)*
- **A Case Study of Taking AP Computer Science Principles (AP CSP): A Student's Perspective** 1588
  Sarah Cameron *(Department of Computer Science, University of West Florida)*,
  Tony Pham *(Department of Computer Science, University of West Florida)*,
  Sikha Bagui *(Department of Computer Science, University of West Florida)*
- **Exploring Collaboration as a Bridge to K-8 CS Integration** 1590
  Justin Cannady *(Northern Lights Collaborative for Computing Education, University of Minnesota)*,
  Jennifer Rosato *(Northern Lights Collaborative for Computing Education, University of Minnesota)*,
  Paul Schonfeld *(Northern Lights Collaborative for Computing Education, University of Minnesota)*

- **On a Path to Indigenization and Decolonization of Computer Science Teaching** 1592
  Sarah Carruthers *(Vancouver Island University*
- **Embracing Mathematical Conjecture Through Coding and Computational Thinking** 1594
  Sarah D. Castle *(University of Idaho)*,
- **Enhancing Automated Feedback in On-Going Assignments** 1596
  Huanyi Chen *(Department of Electrical and Computer Engineering, University of Waterloo)*,
  Paul A.S. Ward *(Department of Electrical and Computer Engineering, University of Waterloo)*
- **Assessment-via-Teaching: Exploring an Alternative Assessment Strategy in Undergraduate Introductory Data Science Course** 1598
  Lujie Karen Chen *(University of Maryland, Baltimore County)*,
  Justin Thai *(University of Maryland, Baltimore County)*
- **Pair Programming with ChatGPT** 1600
  Xi Chen *(Utah Valley University)*, Jingsai Liang *(Westminster University)*
- **Advancing Equity through a Research-Practice Partnership: A Case Study of a Professional Development Initiative for Educators of Color** 1602
  Joshua Childs *(Educational Leadership and Policy, University of Texas at Austin)*,
  Kaitlin Ogden *(Education Leadership and Policy, University of Texas at Austin)*,
  Carol Fletcher *(Texas Advanced Computing Center)*,
  Allen Antoine *(Expanding Pathways in Computing & Texas Advanced Computing Center)*,
  Stephanie Baker *(Texas Advanced Computing Center & University of Texas at Austin)*,
  Alejandro Madrigal *(University of Texas at Austin)*
- **Data Science Mastery Learning Using Parsons Problems-Inspired Table Transformations** .. 1604
  Jacob Seungwon Choe *(UC Berkeley)*, Matthew G. Lee *(UC Berkeley)*, Siddharth A. Marathe *(UC Berkeley)*, Armando Fox *(UC Berkeley)*, Dan Garcia *(UC Berkeley)*, Narges Norouzi *(UC Berkeley)*
- **Dynamic, Animated Feedback for Randomized Problems with Computer-Based Testing** 1606
  Arihant Choudhary *(University of California, Berkeley)*, Dan Garcia *(University of California, Berkeley)*
- **Champion, Collaborator, Curator, Anchor: Supporting K-12 Teachers to Integrate Computational Thinking** 1608
  Merijke Coenraad *(Learning Experience Design, Digital Promise Global)*,
  Alessandra Rangel *(Learning Experience Design, Digital Promise Global)*,
  Quinn Burke *(Learning Experience Design, Digital Promise Global)*,
  Kelly Mills *(Learning Experience Design, Digital Promise Global)*,
  Pati Ruiz *(Learning Sciences Research, Digital Promise Global)*,
  Kyle Dunbar *(Learning Experience Design, Digital Promise Global)*
- **Student Preconceptions of Artificial Intelligence: Results from Single Institution Survey** 1610
  Noah Q. Cowit *(University of Colorado Boulder)*, Casey Fiesler *(University of Colorado Boulder)*
- **Exploring Technology for Accessibility Through a First-Year Writing Seminar** 1612
  Pamela Cutter *(Computer Science Department, Kalamazoo College)*
- **Centering Ethics in Computing Student's Mind: A Question Directed Approach for Teaching and Learning Tech Ethics** 1614
  Udayan Das *(Mathermatics and Computer Science, Saint Mary's College of California)*
- **Reducing Barriers to Entry by Removing Prerequisites for a CS1 Introductory Programming Course** 1616
  Udayan Das *(Mathermatics and Computer Science, Saint Mary's College of California)*,
  Chris Fulton *(School of Continuing and Professional Studies, Loyola University Chicago)*
- **An LLM-based Reflection Analysis Tool for Identifying and Addressing Challenging Topics** 1618
  Nasrin Dehbozorgi *(Kennesaw State University)*, Mourya T. Kunuku *(Kennesaw State University)*
- **Introducing Theoretical Computer Science Education in Social Sciences and Economics Degrees** 1620
  Rafael del Vado Vírseda *(Universidad Complutense de Madrid)*

- **Exploring Student Misconceptions about Concurrency Using the Domain-Specific Programing Language "Sonic Pi"** 1622
  Giorgio Delzanno *(DIBRIS, Università di Genova)*, Giovanna Guerrini *(DIBRIS, Università di Genova)*, Daniele Traversaro *(DIBRIS, Università di Genova)*
- **Style Anomalies Can Suggest Cheating in CS1 Programs** 1624
  Benjamin Denzler *(Computer Science & Engineering, University of California, Riverside)*, Frank Vahid *(Computer Science & Engineering, University of California, Riverside)*, Ashley Pang *(Computer Science and Engineering, University of California, Riverside)*
- **Creation of a CS1 Course with Modern C++ Principles** 1626
  Ryan E. Dougherty *(United States Military Academy)*
- **Designing Theory of Computing Backwards** 1628
  Ryan E. Dougherty *(United States Military Academy)*
- **Solar Weather, Simulation, and AI in Middle School: Developing a Case Study** 1630
  Esther Douglass *(Hillsdale Public Schools)*, Katherine G. Herbert *(School of Computing, Montclair State University)*, Vaibhav Anu *(School of Computing, Montclair State University)*, Thomas J. Marlowe *(Department of Computer Science, Seton Hall University)*, Sumi Hagiwara *(Montclair State Universtiy)*
- **Analogies in Upper Division Computer Science Courses** 1632
  Maria R. Ebling *(United States Military Academy)*, Ryan E. Dougherty *(United States Military Academy)*, Nicholas J. Clark *(United States Military Academy)*
- **The Relationships Between Modality, Peer Instruction Discussion, and Class Sentiment in Hybrid Courses** 1634
  Salma El Otmani *(Duke University)*, Janet Jiang *(Duke University)*, Shao-Heng Ko *(Duke University)*, Kristin Stephens-Martinez *(Duke University)*
- **Hiring, Training, and Managing Undergraduate Teaching Assistants for Large CS1 Classes** 1636
  Megan Englert *(University of Colorado Boulder)*, Lecia Barker *(University of Colorado Boulder)*, Austin Cory Bart *(University of Delaware)*
- **"Their religion is yoga": Understanding the Role of Spirituality for Black Women in Computing** 1638
  Miracle Etim-Andy *(University of Maryland)*, Sheena Erete *(University of Maryland)*, Yolanda A. Rankin *(Emory University)*
- **Enhancing Code Tracing Question Generation with Refined Prompts in Large Language Models** 1640
  Aysa X. Fan *(University of Illinois at Urbana-Champaign)*, Rully A. Hendrawan *(University of Pittsburgh)*, Yang Shi *(North Carolina State University)*, Qianou Ma *(Carnegie Mellon University)*
- **Generalizing the CS Course-based Undergraduate Research Experience (CURE)** 1642
  Amanda S. Fernandez *(University of Texas at San Antonio)*
- **SQLearn: Automated SQL Statement Assessment using Structure-based Analysis** 1644
  Sumukhi Ganesan *(Khoury College of Computer Sciences, Northeastern University)*, Tianchou Gong *(Khoury College of Computer Sciences, Northeastern University)*, Jeongkyu Lee *(Khoury College of Computer Sciences, Northeastern University)*
- **Using Survival Analysis to Model Students' Patience in Online Office Hour Queues** 1646
  Zhikai Gao *(North Carolina State University)*, Adam Gaweda *(North Carolina State University)*, Collin Lynch *(North Carolina State University)*, Sarah Heckman *(North Carolina State University)*, Damilola Babalola *(North Carolina State University)*, Gabriel Silva de Oliveira *(North Carolina State University)*
- **Restorying with AI Art among Latinx Elementary Students** 1648
  Leiny Garcia *(University of California, Irvine)*, Santiago Ojeda-Ramirez *(University of California, Irvine)*, Mark Warschauer *(University of California, Irvine)*
- **Augmenting Computer Science Undergraduate TA Training Programs to Benefit Recruitment, Refinement, and Retention** 1650
  Julie V. Gawenda *(University of Illinois Chicago)*, Adam T. Koehler *(University of Illinois Chicago)*, Agne Nakvosaite *(University of Illinois Chicago)*

- **Leveraging Kotter's 8 Stage Model of Organizational Change to Understand Broadening Participation in Computing** 1652
  Kari L. George *(Department of Computer Science, University of Illinois Urbana-Champaign)*, Max Fowler *(Department of Computer Science, University of Illinois Urbana-Champaign)*, Vidushi Ojha *(Department of Computer Science, University of Illinois Urbana-Champaign)*, Morgan M. Fong *(Department of Computer Science, University of Illinois Urbana-Champaign)*, Kathleen Isenegger *(Department of Computer Science, University of Illinois Urbana-Champaign)*, Christopher Perdriau *(Department of Computer Science, University of Illinois Urbana-Champaign)*, Mariam Saffar Perez *(Department of Computer Science, University of Illinois Urbana-Champaign)*, Yael Gertner *(Department of Computer Science, University of Illinois Urbana-Champaign)*, Colleen M. Lewis *(Department of Computer Science, University of Illinois Urbana-Champaign)*
- **Belonging in POGIL Gateway Classes** 1654
  Olga Glebova *(University of Connecticut)*, Charity Lovitt *(University of Washington Bothell)*, Giselle Mak *(University of Washington Bothell)*, Darian Castillo *(University of Washington Bothell)*, Matthew Horn *(Utah Valley University)*, Tracey Murray *(Capital University)*
- **CAET: Code Analysis and Education Tutor** 1656
  Colin Glynn *(Computer Science Department, College of Saint Benedict and Saint John's University)*, Emily Hed *(Computer Science Department, College of Saint Benedict and Saint John's University)*, Abbigail Pexa *(Computer Science Department, College of Saint Benedict and Saint John's University)*, Tyler Pohlmann *(Computer Science Department, College of Saint Benedict and Saint John's University)*, Imad Rahal *(Computer Science Department, College of Saint Benedict and Saint John's University)*, Robert Hesse *(Computer Science Department, College of Saint Benedict and Saint John's University)*
- **Increasing Diversity of Computing Course Participation Using a Summer Bridge Program** 1658
  Elias Gonzalez *(Computer Science, University of Maryland)*, Kathryn Atchison *(University of Maryland)*
- **Engaging Students from Rural Communities in AI Education with Game-Based Learning** .. 1660
  Alex Goslen *(North Carolina State University)*, Anisha Gupta *(North Carolina State University)*, Smrithi Muthukrishnan *(North Carolina State University)*, Raven Midgett *(North Carolina State University)*, Wookhee Min *(North Carolina State University)*, Jessica Vandenberg *(North Carolina State University)*, Veronica Cateté *(North Carolina State University)*, Bradford Mott *(North Carolina State University)*
- **Supporting Instructors Adoption of Peer Instruction** 1662
  Xingjian Gu *(University of Michigan)*, Barbara J. Ericson *(University of Michigan)*, Zihan Wu *(University of Michigan)*
- **A Study of Undergraduate Learning Assistants (ULAs) in Computer Science** 1664
  Shirin Haji Amin Shirazi *(University of California, Riverside)*, Mariam Salloum *(University of California, Riverside)*, Neftali Watkinson *(University of California, Riverside)*
- **Generative AI as a Resource for Creativity in Computational Physics** 1666
  Patti C. Hamerski *(Oregon State University)*
- **A Systematic Literature Mapping of COVID-19 Papers in Computer Science Education** 1668
  Brian Harrington *(University of Toronto Scarborough)*, Sharon Alex *(University of Toronto Scarborough)*, Leon Lee *(University of Toronto Scarborough)*, Colin Lin *(University of Toronto Scarborough)*, Zixiao Ren *(University of Toronto Scarborough)*, Youxin Tan *(University of Toronto Scarborough)*, Conroy Trinh *(University of Toronto Scarborough)*, Shengsong Xu *(University of Toronto Scarborough)*, Austin Yang *(University of Toronto Scarborough)*
- **Alternative Evaluation in CS Education Research: A Systematic Literature Map** 1670
  Brian Harrington *(University of Toronto Scarborough)*, Thezyrie Amarouche *(University of Toronto Scarborough)*, Andrew Aucie *(University of Toronto Scarborough)*, Shreeansha Bhattarai *(University of Toronto Scarborough)*, Yuxin Chen *(University of Toronto Scarborough)*, Raha Gharadaghi *(University of Toronto Scarborough)*, Kaiyi Huang *(University of Toronto Scarborough)*, Jin Jiang *(University of Toronto Scarborough)*, Linda Lo *(University of Toronto Scarborough)*, Maliha Lodi *(University of Toronto Scarborough)*, Rohita Nalluri *(University of Toronto Scarborough)*, Fawaz Omidiya *(University of Toronto Scarborough)*, Anagha Vadarevu *(University of Toronto Scarborough)*
- **Towards Reuse of Resources and Replication of Results to Improve Instruction on Refactoring Conditionals for Novice Programmers** 1672
  Cecily Heiner *(Computer Systems Engineering Technology, Oregon Institute of Technology)*, Dana Nelson *(Computer Systems Engineering Technology, Oregon Institute of Technology)*

- **Support Patterns in Classrooms Implementing a Computer Science and Physical Computing Curriculum** 1674
  Nga Hoang *(School of Education, University of Colorado Boulder)*,
  Jeffrey B. Bush *(Institute of Cognitive Science, University of Colorado Boulder)*,
  Indrani Dey *(Educational Psychology, University of Wisconsin-Madison)*
- **Incorporating Retakes in a Robotics Class with Labs** 1676
  Janine Hoelscher *(University of North Carolina at Chapel Hill)*,
  Bryce Ikeda *(University of North Carolina at Chapel Hill)*,
  Daniel Szafir *(University of North Carolina at Chapel Hill)*,
  Ron Alterovitz *(University of North Carolina at Chapel Hill)*
- **Thoughtfully Designing an Industry Mentor Program for Undergraduates in CS and Mathematics** 1678
  Victoria Hong *(St. Joseph's University)*, Cheyne Glass *(State University of New York at New Paltz)*
- **Towards Attention-Based Automatic Misconception Identification in Introductory Programming Courses** 1680
  Muntasir Hoq *(North Carolina State University)*, Jessica Vandenberg *(North Carolina State University)*,
  Bradford Mott *(North Carolina State University)*, James Lester *(North Carolina State University)*,
  Narges Norouzi *(University of California Berkeley)*, Bita Akram *(North Carolina State University)*
- **Analyzing Student Performance with Free Late Submission Days** 1682
  John R. Hott *(University of Virginia)*
- **Towards More Efficient Office Hours for Large Courses: Using Cosine Similarity to Efficiently Construct Student Help Groups** 1684
  John R. Hott *(University of Virginia)*, Mark Floryan *(University of Virginia)*,
  Nada Basit *(University of Virginia)*
- **Integrating Personalized Parsons Problems with Multi-Level Textual Explanations to Scaffold Code Writing** 1686
  Xinying Hou *(University of Michigan)*, Barbara J. Ericson *(University of Michigan)*,
  Xu Wang *(University of Michigan)*
- **Building a Mixed-format Computer Science Assessment for Middle School** 1688
  Jennifer Houchins *(WestEd)*, Kim Luttgen *(WestEd)*, Rosalind Owen *(WestEd)*,
  Lydia Martinez Rivera *(WestEd)*, Matt Silberglitt *(WestEd)*, Yvonne Kao *(WestEd)*
- **Programming Language Knowledge Transfer that Teachers Observe in their Classrooms** .. 1690
  Jennifer Houchins *(WestEd)*, Rosalind Owen *(WestEd)*, Bryan Matlen *(WestEd)*,
  Yvonne Kao *(WestEd)*
- **Enhancing Formative Feedback at Scale with the Intelligent Feedback Assistant** 1692
  Rifa Jamal *(University of New South Wales)*, Jake Renzella *(University of New South Wales)*
- **Striking a Chord through a Mixed-Methods Study of Music-Based Learning to Leverage Music and Creativity to Bridge the Gender Gap in Computer Science** 1694
  Fatemeh Jamshidi *(Computer Science and Software Engineering, Auburn University)*,
  Maryam Bigonah *(Computer Science and Software Engineering, Auburn University)*,
  Daniela Marghitu *(Computer Science and Software Engineering, Auburn University)*
- **Root the (Ballot) Box: Designing Security Engineering Courses with E-Voting** 1696
  Tushar M. Jois *(City College of New York)*, Atheer Almogbil *(Johns Hopkins University)*,
  Logan Kostick *(Johns Hopkins University)*
- **Bridging the Gap: Computational Thinking as a Pathway to Equitable Math Education for Neurodiverse Learners** 1698
  Kate Kastelein *(Maine Mathematics and Science Alliance)*,
  Kelsie Dawson *(Maine Mathematics and Science Alliance)*,
  Rhonda Tate *(Maine Mathematics and Science Alliance)*
- **Computer Science Kickstart: An Innovative Bootcamp to Ignite Passion in First-Year Female-Identifying University Students** 1700
  Stella Kaval *(Electrical Engineering and Computer Science, University of California, Berkeley)*,
  Mallika Reddy *(Electrical Engineering and Computer Science, University of California, Berkeley)*,
  Abigail O'Neill *(Electrical Engineering and Computer Science, University of California, Berkeley)*,

- **A Family of Instruments to Measure Data Science Attitudes** 1702
  April Kerby-Helm *(Department of Mathematics and Statistics, Winona State University)*,
  Michael A. Posner *(Department of Mathematics and Statistics, Villanova University)*,
  Alana Unfried *(California State University, Monterey Bay)*,
  Douglas Whitaker *(Department of Mathematics and Statistics, Mount Saint Vincent University)*,
  Marjorie E. Bond *(Department of Statistics, Pennsylvania State University)*,
  Leyla Batakci *(Department of Mathematical Science, Elizabethtown College)*
- **An Experience Report on Teaching a Large Introductory Programming Course in Hybrid Mode** 1704
  Shyama Sastha Krishnamoorthy Srinivasan *(IIIT-Delhi)*, Avi Gupta *(IIIT-Delhi)*, Pankaj Jalote *(IIIT-Delhi)*
- **Scaffolded Projects for the Social Good: A Strategy for Deploying Studio Model in CS Education** 1706
  Stan Kurkovsky *(Central Connecticut State University)*, Mikey Goldweber *(Denison University)*,
  Nathan Sommer *(Xavier University)*, Chad A. Williams *(Central Connecticut State University)*
- **Poprawa: A Gradebook Library for Alternatively-Graded Courses** 1708
  Zachary Kurmas *(Grand Valley State University)*
- **Exploring AIFORGOOD Summer Camp Curriculum to Foster Middle School Students' Understanding of Artificial Intelligence** 1710
  Kyungbin Kwon *(Indiana University)*, Keunjae Kim *(Indiana University)*,
  Anne T. Ottenbreit-Leftwich *(Indiana University)*, Krista D. Glazewski *(North Carolina State University)*,
  Matthew L. Brown *(Indiana University)*, Haesol Bae *(State University of New York)*,
  Florentina M. Closser *(Indiana University)*
- **Facilitating Teens as Ethical Sensemakers of Technology** 1712
  Rotem Landesman *(University of Washington)*, Jean Salac *(University of Washington)*,
  Amy J. Ko *(University of Washington)*
- **DCC Sidekick: Helping Novices Solve Programming Errors Through a Conversational Explanation Interface** 1714
  Lorenzo Lee Solano *(University of New South Wales)*, Jake Renzella *(University of New South Wales)*,
  Alexandra Vassar *(University of New South Wales)*
- **Why Learn This? Visualizing Pathways between CS Course Topics and Careers** 1716
  Stacey Levine *(Computer Science, Georgia State University)*,
  Anu G. Bourgeois *(Computer Science, Georgia State University)*
- **Supporting Student Engagement in K-12 AI Education with a Card Game Construction Toolkit** 1718
  Hansol Lim *(North Carolina State University)*, Wookhee Min *(North Carolina State University)*,
  Jessica Vandenberg *(North Carolina State University)*, Veronica Cateté *(North Carolina State University)*,
  Judith Uchidiuno *(Georgia Institute of Technology)*, Bradford Mott *(North Carolina State University)*
- **Increasing Access to CS Instruction in Low-Income Afterschool Settings** 1720
  Jared Lim *(Georgia Institute of Technology)*, Grace Barkhuff *(Georgia Institute of Technology)*,
  Judith Uchidiuno *(Georgia Institute of Technology)*
- **Teaching Algorithm Design: A Literature Review** 1722
  Jonathan Liu *(University of Chicago)*, Seth Poulsen *(Utah State University)*,
  Hongxuan Chen *(University of Illinois at Urbana-Champaign)*, Grace Williams *(University of Chicago)*,
  Yael Gertner *(University of Illinois at Urbana-Champaign)*, Diana Franklin *(University of Chicago)*
- **The Current Research Landscape of Computing Education in Elementary Settings: A Systematic Literature Review** 1724
  Ruohan Liu *(University of Virginia)*
- **Circuit Tiles: Mixing Culture, Crafts, and Circuits to Inspire Diversity in Computing** 1726
  Maria Lopez-Delgado *(Information Systems, University of Maryland, Baltimore County)*
- **Leveraging Science Fiction to Prepare Students for Tech Battles** 1728
  Jessica Lucchetta *(Department of Information Engineering and Computer Science, University of Trento)*,
  Andrea Saiani *(Witted Srl)*, Milena Stoycheva *(Junior Achievement Bulgaria)*,
  Lorenzo Angeli *(Department of Information Engineering and Computer Science, University of Trento)*

- **Evaluating Storytelling Videos Using YouTube Analytics** 1730
  Anna Ly *(University of Toronto Mississauga)*, Tingting Zhu *(University of Toronto Mississauga)*, Andrew Petersen *(University of Toronto Mississauga)*
- **Computer Science Curriculum Trends** 1732
  Sean Mackay *(Department of Engineering Education, University at Buffalo)*, Adrienne Decker *(Department of Engineering Education, University at Buffalo)*
- **Developing Interactive Exercise Materials for Machine Learning Using Spreadsheets** 1734
  Atsuhiko Maeda *(Kanagawa Institute of Technology)*
- **A Generalized Framework for Describing Question Randomization** 1736
  Romina Mahinpei *(Computer Science, The University of British Columbia)*, Iris Xu *(Computer Science, The University of British Columbia)*, Steven Wolfman *(Computer Science, The University of British Columbia)*, Firas Moosvi *(Computer Science, The University of British Columbia)*
- **Mastery with Method: Calibrating Policies to Boost Completion and Sentiment in a Computing Course using Mastery Learning** 1738
  Vedansh Malhotra *(UC Berkeley)*, Jenny Mendez Mendez *(UC Berkeley)*, Daniel Garcia *(UC Berkeley)*
- **Perception, Trust, Attitudes, and Models: Introducing Children to AI and Machine Learning with Five Software Exhibits** 1740
  Fred Martin *(University of Texas at San Antonio)*, Saniya Vahedian Movahed *(University of Texas at San Antonio)*, James Dimino *(University of Massachusetts Lowell)*, Andrew Farrell *(University of Massachusetts Lowell)*, Elyas Irankhah *(University of Massachusetts Lowell)*, Srija Ghosh *(University of Massachusetts Lowell)*, Garima Jain *(University of Massachusetts Lowell)*, Vaishali Mahipal *(University of Massachusetts Lowell)*, Pranathi Rayavaram *(University of Massachusetts Lowell)*, Ismaila Temitayo Sanusi *(University of Eastern Finland)*, Erika Salas *(University of Massachusetts Lowell)*, Kelilah Wolkowicz *(University of Massachusetts Lowell)*, Sashank Narain *(University of Massachusetts Lowell)*
- **Impacts of Academic Preparedness on CS1 Performance** 1742
  Barbara Martinez Neda *(University of California, Irvine)*, Flor Morales *(University of California, Irvine)*, Kitana Carbajal Juarez *(University of California, Irvine)*, Jennifer Wong-Ma *(University of California, Irvine)*, Sergio Gago-Masague *(University of California, Irvine)*
- **More than a Checklist: Redesigning a UI/UX Curriculum to Emphasize Accessibility** 1744
  Talie Massachi *(Brown University)*
- **Developing Computational Thinking in Middle School Music Technology Classrooms** 1746
  Lauren McCall *(Center for Music Technology, Georgia Institute of Technology)*, Brittney Allen *(Center for Music Technology, Georgia Institute of Technology)*, Jason Freeman *(Center for Music Technology, Georgia Institute of Technology)*, Stephen Garrett *(Center for Music Technology, Georgia Institute of Technology)*, Sabrina Grossman *(Center for Education Integrating Science, Math, and Computing, Georgia Institute of Technology)*, Jed Paz *(Center for Education Integrating Science, Math, and Computing, Georgia Institute of Technology)*, Doug Edwards *(Center for Education Integrating Science, Math, and Computing, Georgia Institute of Technology)*, Tom McKlin *(The Findings Group)*, Taneisha Lee *(The Findings Group)*
- **Unintentional Barriers for AP Computer Science Principles: A Course Designed for Every Student** 1748
  Emma L. McDaniel *(Georgia State University)*, Aaja Christie *(Georgia State University)*, Anu G. Bourgeois *(Georgia State University)*
- **Validation of a Bebras-Based Test to Assess Computational Thinking Abilities in First-Year College Students** 1750
  Federico Meza *(Univ. Técnica Federico Santa María)*, Andrea Vásquez *(Univ. Técnica Federico Santa María)*, Daniel San Martín *(Univ. Técnica Federico Santa María)*
- **Elevating Learning Experiences: Leveraging Large Language Models as Student-Facing Assistants in Discussion Forums** 1752
  Chancharik Mitra *(University of California, Berkeley)*, Mihran Miroyan *(University of California, Berkeley)*, Rishi Jain *(University of California, Berkeley)*, Vedant Kumud *(University of California, Berkeley)*, Gireeja Ranade *(University of California, Berkeley)*, Narges Norouzi *(University of California, Berkeley*

- **A Game to Share the Story of LGBTQ+ Pioneers and Influential Computer Scientists** 1754
  Karina Mochetti *(UBC)*
- **Advice for the First Time Hardware CTF Organizer** 1756
  Chad Mourning *(Ohio University)*
- **Implementation and Evaluation of Equitable Practices in an Introductory CS Course** 1758
  Nadia Najjar *(Software and Information Systems, University of North Carolina at Charlotte)*,
  Debarati Basu *(Engineering Fundamentals, Embry-Riddle Aeronautical University)*
- **Incorporating Technical Interview Preparation into an HBCU Pre-Freshman Programming Bootcamp** 1760
  Whitney Nelson *(Broadening Participation in Computing, Morehouse College)*,
  Kinnis Gosha *(Broadening Participation in Computing, Morehouse College)*,
  Talitha Washington *(AUC Data Science Initiative, Atlanta University Center Consortium)*,
  Nina Gilbert *(The Morehouse Center for Excellence in Education, Morehouse College)*
- **Use of Large Language Models for Extracting Knowledge Components in CS1 Programming Exercises** 1762
  Rose Niosuha *(University of California, Berkeley)*, Muntasir Hoq *(North Carolina State University)*,
  Bita Akram *(North Carolina State University)*, Narges Norouzi *(University of California, Berkeley*
- **Data for Healthy Communities: A Public Interest Pilot Course Designed to Develop K-12 Data Literacy** 1764
  Emily Nutwell *(The Ohio State University)*, Kelsey Badger *(The Ohio State University)*,
  Jessica T. Kulp *(The Ohio State University)*
- **Python Programming Education with Semantics-oriented Screen Reading for K-12 Students with Vision Impairments** 1766
  God'Salvation F. Oguibe *(The University of Texas at San Antonio)*,
  Lauryn M. Castro *(The University of Texas at San Antonio)*,
  Katherine Cantrell *(The University of Texas at San Antonio)*,
  Kathy B. Ewoldt *(The University of Texas at San Antonio)*,
  Leslie Cockerill Neely *(The University of Texas at San Antonio)*,
  Wei Wang *(The University of Texas at San Antonio)*
- **Motivating and Increasing the Relevance of Elementary Computer Science in Rural Communities: Exploring Elementary Teacher Perspectives in Idaho** 1768
  Diana Dajung Oh *(American Institutes for Research)*, Kathryn M. Rich *(American Institutes for Research)*,
  Kathleen L. Bosch-Wilson *(Idaho STEM Action Center)*,
  Heather L. Cunningham *(BootUp Professional Development)*
- **Web-based Course Assessment System of Student Learning Outcomes: A Pilot Study** 1770
  Daniel Ojeda *(Department of Computer Science and Technology, Kean University)*,
  Patricia Morreale *(Department of Computer Science and Technology, Kean University)*,
  Daehan Kwak *(Department of Computer Science and Technology, Kean University)*
- **Blurring the Boundaries between Science, Math and Computer Science** 1772
  Chris Orban *(Ohio State University)*
- **Understanding the Leadership Structure and Mentoring Model of an Extracurricular Robotics Team: Key Findings from a Case Study** 1774
  Hitender K. Oswal *(University of Washington)*, Jonathan Wick *(Western Washington University)*,
  Seth Tandon *(University of Southern California)*, Ashley Brewster *(Science and Math Institute)*,
  Sushil K. Oswal *(University of Washington)*
- **Linguistic Pedagogical Approaches to Transfer in Computer Science** 1776
  Rosalind Owen *(WestEd)*
- **Almost Correct Solvers for Combinational Logic Assignments** 1778
  Sunjae Park *(Wentworth Institute of Technology)*
- **The Potential of Humanitarian Applications to Increase Interest and Motivation of Underrepresented Student Groups** 1780
  Lori Postner *(Nassau Community College)*, Heidi J. C. Ellis *(Western New England University)*,
  Gregory W. Hislop *(Drexel University)*, Wesley Shumar *(Drexel University)*

- **Immersing Undergraduates in Interdisciplinary Course Collaborations** 1782
  S. Monisha Pulimood *(Computer Science, The College of New Jersey)*,
  Diane C. Bates *(Sociology, The College of New Jersey)*,
  Kim Pearson *(Journalism & Prof. Writing, The College of New Jersey)*
- **Student Perceptions of Authentic Learning to Learn White-box Testing** 1784
  Akond Rahman *(Department of Computer Science and Software Engineering, Auburn University)*,
  Yue Zhang *(Department of Computer Science and Software Engineering, Auburn University)*,
  Fan Wu *(Department of Computer Science, Tuskegee University)*,
  Hossain Shahriar *(Center for Cybersecurity, University of West Florida)*
- **Analyzing State-Level High School CS Teacher Certification Through Dataset Exploration** 1786
  Sujeeth Goud Ramagoni *(Marquette University)*, Dennis Brylow *(Marquette University)*
- **Building Awareness of Computational Thinking Pathways across a Large School District** .. 1788
  Alessandra Rangel *(Learning Experience Design, Digital Promise Global)*,
  Merijke Coenraad *(Learning Experience Design, Digital Promise Global)*,
  Pati Ruiz *(Learning Sciences Research, Digital Promise Global)*,
  Kyle Dunbar *(Learning Experience Design, Digital Promise Global)*,
  Lisa Milenkovic *(Applied Learning Department, STEM and Computer Science Program, Broward County Public Schools)*,
  Sheryl Arriola *(Applied Learning Department, STEM and Computer Science Program, Broward County Public Schools)*,
  Annmargareth Marousky *(Applied Learning Department, STEM and Computer Science Program, Broward County Public Schools)*
- **Student (Self) Assessment, Feedback, and Relatability in CS-2 Game Programming Assignments** 1790
  Veena Ravishankar *(University of Mary Washington)*, Jessica Zeitz *(University of Mary Washington)*
- **Experience Helps, but It Isn't Everything: Exploring Causes of Affective State in Novice Programmers** 1792
  Heidi Reichert *(North Carolina State University)*, Sandeep Sthapit *(North Carolina State University)*,
  Benyamin T. Tabarsi *(North Carolina State University)*, Ally Limke *(North Carolina State University)*,
  Thomas Price *(North Carolina State University)*, Tiffany Barnes *(North Carolina State University)*
- **Using Concept Maps for Notional Machine Selection in CS1** 1794
  Ethan Richards *(Colorado School of Mines)*, Sonia Spindt *(Cherry Creek Schools)*,
  Gabriel Fierro *(Colorado School of Mines)*
- **Becoming Core: Curriculum Planning Tools for Integrating CS into K-5 Content Areas** 1796
  Darcy M. Ronan *(College of Education & Human Development, Sacred Heart University)*,
  Heidi Williams *(Marquette University)*
- **Scaling Collaborative Learning: Using the Community Digital Library to Enrich Course Content** 1798
  Kevin Ros *(Computer Science, University of Illinois Urbana-Champaign)*,
  ChengXiang Zhai *(Computer Science, University of Illinois Urbana-Champaign)*
- **Using Generative AI to Support PK-12 Teaching and Learning: Developing Sample Lessons and More** 1800
  Pati Ruiz *(Learning Sciences Research, Digital Promise)*,
  Alessandra Rangel *(Learning Experience Design, Digital Promise)*,
  Merijke Coenraad *(Learning Experience Design, Digital Promise)*
- **Community Building and Joyful Learning in Computer Science Education Through Online At-Scale Seminars** 1802
  Ana Mary Rusch *(Georgia Institute of Technology)*, Eric Ianni *(Georgia Institute of Technology)*
- **Middle School CS Curriculum and Standards Alignment** 1804
  Daisy Rutstein *(edCount, LLC)*, Satabdi Basu *(SRI International)*, Hui Yang *(SRI International)*,
  Arif Rachmatullah *(SRI International)*, Carol Tate *(SRI International)*
- **Elementary Latinx Students Apply Growth Mindset while Creating in Scratch** 1806
  Dana Saito-Stehberger *(University of California, Irvine)*, Jennifer K. Houchins *(WestEd)*,
  Mark Warschauer *(University of California, Irvine)*

- **Programming Assignment Ungrading as a License to Learn: Implementing Specifications Grading in the Undergraduate Web Development Classroom** 1808
  Raghav V. Sampangi *(Faculty of Computer Science, Dalhousie University)*,
  Eric Poitras *(Faculty of Computer Science, Dalhousie University)*,
  Mayra D. Barrera Machuca *(Faculty of Computer Science, Dalhousie University)*
- **Automated Support for Flexible Extensions** 1810
  Jordan Schwartz *(University of California Berkeley)*, Madison Bohannan *(University of California Berkeley)*,
  Jacob Yim *(University of California Berkeley)*, Yuerou Tang *(University of California Berkeley)*,
  Dana Benedicto *(University of California Berkeley)*, Charisse Liu *(University of California Berkeley)*,
  Armando Fox *(University of California Berkeley)*, Lisa Yan *(University of California Berkeley)*,
  Narges Norouzi *(University of California Berkeley)*
- **Barriers to Equitable CS Expansion: Perceptions of Secondary Teachers of Computer Science during COVID-19** 1812
  Robert Schwarzhaupt *(American Institutes for Research)*, Ryan Torbey *(American Institutes for Research)*,
  Bridina Lemmer *(American Institutes for Research)*, Joseph P. Wilson *(American Institutes for Research)*
- **Supporting Teacher Understanding of Computational Thinking Integration into Early Elementary Curricula** 1814
  Heather Sherwood *(Education Development Center)*, Alice Kaiser *(Education Development Center)*,
  Camille Ferguson *(Education Development Center)*, Anthony Negron *(New York Hall of Science)*,
  Ray Ferrer *(New York Hall of Science)*, Don Labonte *(Participate)*
- **CT Readiness for All: Studying a Framework for Supporting Schoolwide Computational Thinking Integration Across Elementary Curricula** 1816
  Heather Sherwood *(Education Development Center)*, Babette Moeller *(Education Development Center)*,
  Cheri Fancsali *(The Research Alliance for NYC Schools)*
- **Novices' Perceptions of Web-Search and AI for Programming** 1818
  James Skripchuk *(North Carolina State University)*, John Bacher *(North Carolina State University)*,
  Yang Shi *(North Carolina State University)*, Keith Tran *(North Carolina State University)*,
  Thomas Price *(North Carolina State University)*
- **The Landscape of Disability-Related K-12 Computing Education Research** 1820
  Julie M. Smith *(CSEdResearch.org)*, Monica M. McGill *(CSEdResearch.org)*
- **Reimagining CS Courses for High School Students** 1822
  Julie M. Smith *(CSEdResearch.org)*, Bryan Twarek *(CSTA)*, Monica M. McGill *(CSEdResearch.org)*
- **Evaluating Large Language Model Code Generation as an Autograding Mechanism for "Explain in Plain English" Questions** 1824
  David H. Smith IV *(University of Illinois)*, Craig Zilles *(University of Illinois)*
- **Programming Plagiarism Detection with Learner Data** 1826
  Yifan Song *(University of Illinois)*, Yuanxin Wang *(Carnegie Mellon University)*,
  Marshall An *(Carnegie Mellon University)*, Christopher Bogart *(Carnegie Mellon University)*,
  Majd Sakr *(Carnegie Mellon University)*
- **See Logic In Life: An Individualized Exercise Structure for Making and Sharing Computing Concept Connections** 1828
  Madalene Spezialetti *(Computer Science Department, Trinity College)*
- **Cracking the Cultural Code: Understanding the Cultural Barriers for Asian International CS Students in the US** 1830
  Sandeep Sthapit *(North Carolina State University)*, Madison Thomas *(North Carolina State University)*,
  Janet Brock *(North Carolina State University)*, Tiffany Barnes *(North Carolina State University)*
- **ThemeRec: Personalizing IDE Themes for Students** 1832
  Jialiang Tan *(Lehigh University)*, Yu Chen *(William and Mary)*,
  Shuyin Jiao *(North Carolina State University)*
- **Supporting Mastery Learning with Flexible Extensions** 1834
  Yuerou Tang *(University of California Berkeley)*, Jacob Yim *(University of California Berkeley)*,
  Jordan Schwartz *(University of California Berkeley)*, Madison Bohannan *(University of California Berkeley)*,
  Dana Benedicto *(University of California Berkeley)*, Charisse Liu *(University of California Berkeley)*,
  Armando Fox *(University of California Berkeley)*, Lisa Yan *(University of California Berkeley)*,
  Narges Norouzi *(University of California Berkeley)*

- **Learning Software Engineering Principles with Program Wars v.3.0** 1836
  Md. Hasan Tareque *(University of Lethbridge)*, John Anvik *(University of Lethbridge)*
- **Enhancing Programming Education through Game-Based Learning: Design and Implementation of a Puyo Puyo–Inspired Teaching Tool** 1838
  Ruochen Tian *(Waseda University)*, Daisuke Saito *(Takachiho University)*, Hironori Washizaki *(Waseda University)*, Yoshiaki Fukazawa *(Waseda Univesity)*, Hiroshi Kobayashi *(System Information Co., Ltd.)*, Ayumi Tsuji *(System Information Co., Ltd.)*
- **Using Computational Thinking to Learn Creative Writing and Performance Arts: Summer Experiences of Story Creation Projects** 1840
  Ruth Torres Castillo *(New Mexico State University)*, Tomas Valdez *(New Mexico State University)*, Enrico Pontelli *(New Mexico State University)*, Horacio Gonzalez *(New Mexico State University)*, Erick Nevarez *(New Mexico State University)*, Raena Cota *(New Mexico State University)*
- **Assessing the Efficacy of Goal-Based Scenarios in Scaling AI Literacy for Non-Technical Learners** 1842
  Ying-Jui Tseng *(Carnegie Mellon University)*, Ruiwei Xiao *(Carnegie Mellon University)*, Christopher Bogart *(Carnegie Mellon University)*, Jaromir Savelka *(Carnegie Mellon University)*, Majd Sakr *(Carnegie Mellon University)*
- **Math IDE: A Platform for Creating with Math** 1844
  Sierra Wang *(Stanford University)*, John Mitchell *(Stanford University)*, Nick Haber *(Stanford University)*, Chris Piech *(Stanford University)*
- **Measuring CS Student Attitudes Toward Large Language Models** 1846
  Jason Lee Weber *(University of California, Irvine)*, Barbara Martinez Neda *(University of California, Irvine)*, Kitana Carbajal Juarez *(University of California, Irvine)*, Jennifer Wong-Ma *(University of California, Irvine)*, Sergio Gago-Masague *(University of California, Irvine)*, Hadar Ziv *(University of California, Irvine)*
- **Computational Thinking in STEM Teaching: Preservice Teachers' Conceptualizations and Practices** 1848
  Wei Wei *(University of California, Los Angeles)*, Jesse Ha *(Montclair State University)*, Kimberley Gomez *(University of California, Los Angeles)*, Anuradha Ghosh *(Montclair State University)*
- **Perfect Snake: Exploring Algorithms Through the Snake Game** 1851
  William Weinberger *(University of Southern California)*, Jingsai Liang *(Westminster University)*, Xi Chen *(Utah Valley University)*
- **A Novel Scaffolded Assessment Bridging Concepts and Code** 1853
  Nathaniel Weinman *(University of California, Berkeley)*, Jack Boreczky *(University of California, Berkeley)*, Armando Fox *(University of California, Berkeley)*
- **Discovering Computer Science: A High School CS Course Co-developed by College & High School Teachers** 1855
  Pauline White *(Siena College)*, Robin Flatland *(Siena College)*, James Matthews *(Siena College)*, Jesse Moya *(Siena College)*, Kathryn Schiller *(University at Albany)*
- **Guided Undergraduate Training for Shark Segmentation (GUTSS)** 1857
  Amy Wu *(University of Florida)*, Morgan Cobb *(University of Florida)*, Victor Perez *(Department of Environmental Studies, St. Mary's College of Maryland)*, Christan Grant *(Department of Computer and Information Science and Engineering, University of Florida)*, Jeremy A. Magruder Waisome *(Department of Engineering Education, University of Florida)*
- **Interactive Learning Modules for Fostering Secure Coding Proficiency in Introductory Programming Courses** 1859
  Guangming Xing *(Western Kentucky University)*, Gongbo Liang *(Texas A&M University - San Antonio)*, Tawfiq Salem *(Purdue University)*
- **LEARNDB: A Comprehensive Toolkit for Database Education** 1861
  Guangming Xing *(Western Kentucky University)*, Tawfiq Salem *(Purdue University)*, Gongbo Liang *(Texas A&M University - San Antonio)*
- **A Cross-Case Analysis of Experienced Educators in CS Inclusion** 1863
  Wei Yan *(University of Florida)*, Andrew Bennett *(University of Florida)*, Alexis Cobo *(University of Florida)*, Maya Israel *(University of Florida)*

Special Sessions

- **Advancing Social Justice through Data Science** 1865
  John Bartucz *(College of Education and Human Development, University of Minnesota)*,
  Ana Smaranda Sandu *(Computer Science, Wellesley College)*,
  Laney Strange *(Khoury College of Computer Sciences, Northeastern University)*,
  R.N. Uma *(Mathematics and Physics, North Carolina Central University)*
- **Computer Science Curricula 2023 (CS2023): The Final Report** 1867
  Amruth N. Kumar *(Ramapo College of New Jersey)*, Rajendra K. Raj *(Rochester Institute of Techinology)*
- **Microteaching: Binary Heaps, Side-Channel Attacks, Equitable Grading, Java Classes, Loops, and 3D Java** 1869
  Colleen M. Lewis *(Computer Science Department, Univ. of Illinois Urbana-Champaign)*,
  Cynthia Bailey Lee *(Computer Science Department, Stanford University)*,
  Adam Blank *(Computing & Mathematical Sciences, California Institute of Technology)*,
  Maria Camarena *(Maywood Center for Enriched Studies, Los Angeles Unified School District)*,
  Manuel Hernandez *(Mathematics Department, Bell Gardens High School, MUSD & Math for America - Los Angeles)*,
  Frank Vahid *(Computer Science & Engineering, University of California, Riverside)*
- **Unlocking Excellence in Educational Research: Guidelines for High-Quality Research that Promotes Learning for All** 1871
  Monica M. McGill *(CSEdResearch.org)*, Sarah Heckman *(North Carolina State University)*,
  Michael Liut *(University of Toronto Mississauga)*, Ismaila Temitayo Sanusi *(University of Eastern Finland)*,
  Claudia Szabo *(University of Adelaide)*
- **Selecting BPC Activities: Discussion of the Feasibility, Relevance, and Sustainability of 60+ Activities from BPCnet.org** 1873
  Briana Scurry *(Computer Science Department, Univ. of North Carolina at Chapel Hill)*,
  Colleen M. Lewis *(Computer Science Department, Univ. of Illinois Urbana-Champaign)*,
  Luther A. Tychonievich *(Computer Science Department, Univ. of Illinois Urbana-Champaign)*
- **Innovative Strategies for genAI in CS Courses** 1875
  Jill Westerlund *(Computer Science, University of Alabama)*,
  Sandra Czajka *(Computer Science, Riverside Brookfield High School)*,
  Andrew Kuemmel *(Computer Sciences, University of Wisconsin)*

ACM Student Research Competition Abstracts

- **Roro8: A Fantasy Computer for Computer Architecture and Organization Education** 1877
  Zeyad Ahmed *(Assiut University)*
- **Learning with the Rashomon Augmented Reality Cube (RARc)** 1877
  Candy Arce *(New Mexico State University)*, Angie Hernandez *(New Mexico State University)*,
  Ruth Torres Castillo *(New Mexico State University)*, Tomas Valdez *(New Mexico State University)*,
  Raena Cota *(New Mexico State University)*, Enrico Pontelli *(New Mexico State University)*
- **Ethics vs. Abstraction: Comparing Learning Outcomes from Ethics-Integrated and Technical-Only Instruction** 1878
  Noelle Brown *(University of Utah)*
- **Revolutionizing Student Engagement and Enrollment through Personalized, AI-Driven Dialog Systems in Higher Education** 1879
  Alexander Fox *(Elizabethtown College)*, Joshua Stoner *(Elizabethtown College)*,
  Jingwen Wang *(Elizabethtown College)*
- **Enhancing American Sign Language Classification by Leveraging Hand Landmark Extraction** 1879
  Benjamin Thomas Guerrieri *(The College of New Jersey)*
- **Exploring K-12 Teachers' Trajectories of Engagement with Critically Conscious Computing** 1880
  Brendan Henrique *(University of California, Berkeley)*

- **Accessing and Democratizing AI for Whom? Student Learning through an Algorithm-Centered Supply Chain Case Study** 1880
  Ashish Hingle *(George Mason University)*
- **The Future of the Error Message: Comparing Large Language Models and Novice Programmer Effectiveness in Fixing Errors** 1881
  Brij Howard-Sarin *(Computer Science, Williams College)*
- **A Comparative Study of Few-Shot vs. Zero-Shot Prompting to Generate Quick and Useful Responses to Students' Periodic Reflections** 1881
  Ethan Hurley *(UNC Charlotte)*, Joel Okyere-Badoo *(UNC Charlotte)*
- **Recommendation System Algorithm Application to Increase Interdisciplinary Undergraduate Course Collaborations** 1882
  Julianna Iannini *(The College of New Jersey)*, Kyla Ramos *(The College of New Jersey)*, Joe Petrecca *(The College of New Jersey)*
- **Creating Instructional Elementary Programming Videos for Use in an Adaptive Testing and Remediation System** 1882
  Abigail Jones *(Northern Kentucky University)*, Manogya Aryal *(Northern Kentucky University)*, Justin Selby *(Northern Kentucky University)*, Seth Adjei *(Northern Kentucky University)*
- **Understanding Engineering Students' Ethical and Algorithmic Decision Preferences through a Consequentialist Framework** 1883
  Edward Kempa *(University of Florida)*, Ashish Aggarwal *(University of Florida)*
- **Mining Students' Mastery Levels from CS Placement Tests via LLMs** 1883
  Yi Li *(Boston College)*, Riteng Zhang *(Boston College)*, Danni Qu *(Boston College)*, Maíra Marques Samary *(Boston College)*
- **Simplifying Signaling Pathway Reconstruction with Containerized Random Walk Algorithms** 1884
  Yancheng Liu *(Reed College)*, Anna Ritz *(Reed College)*
- **Energizing Web Development in the Exploring Computer Science Curriculum** 1884
  Samuel B. Mazzone *(Marquette University)*
- **Is ChatGPT the Academic Catalyst We've all been Waiting For?** 1885
  Raneem Emad Mousa *(Simmons University)*, Nanette Veilleux *(Simmons University)*
- **Understanding Undergraduate Students' Participation in Computing Clubs** 1885
  Brooke Nelson *(University of Florida)*, Amanpreet Kapoor *(University of Florida)*, Christina Gardner-Mccune *(University of Florida)*
- **Askers, Answerers, Non-Answerers, and Lurkers: Investigating Online Q&A Platforms Personas and Success in a CS1 Course** 1886
  Nigel Newby *(University of Pennsylvania)*
- **Enhancing Scratch Programming with Autograding** 1886
  Herart Dominggus Nurue *(The University of Alabama)*
- **Cloud-Secure: An Investigation into Firefly and Grey Wolf Optimization Algorithms for Phishing Detection with Machine Learning Classifiers** 1887
  Kelvin Ovabor *(The University of Alabama)*
- **Understanding Outcome Expectancy in a CS1 Course** 1887
  Griffin Pitts *(University of Florida)*, Ashish Aggarwal *(University of Florida)*
- **Edit, Run, Error, Repeat: Learning Analytics to Find the Most Improved Programming Student** 1888
  Johan Mattias Snider *(Uppsala University)*
- **Undergraduate Facilitators as Role Models for Middle School Learners within an AI Summer Camp** 1888
  Carly Solomon *(University of Florida)*
- **Towards a Verified Implementation of a Network Stack** 1889
  Caitlyn Wilde *(Lewis and Clark College)*, Wyeth T. Greenlaw Rollins *(Lewis and Clark College)*, Levi Overcast *(Evergreen State College)*, Alain Kägi *(Lewis and Clark College)*

- **A Web Development and Cloud Deployment Framework for a Software Engineering Course** 1890
  Michael Wilkinson *(Worcester Polytechnic Institute)*, Ariel Schechter *(Worcester Polytechnic Institute)*, Bryce Lukens *(Worcester Polytechnic Institute)*, Ian Wright *(Worcester Polytechnic Institute)*, Joseph Cardarelli *(Worcester Polytechnic Institute)*
- **SQL Query Evaluation with Large Language Model and Abstract Syntax Trees** 1890
  Lili Xiang *(Khoury College of Computer Science, Northeastern University)*
- **Understanding Support Needs in Computer Science Labs** 1891
  Weiyin Xu *(University of San Diego)*

SIGCSE TS 2024 Workshops

- **Hands-on Accessibility Teaching T raining for Computer Science Instructors** 1892
  Catherine M. Baker *(Creighton University)*, Yasmine N. Elglaly *(Western Washington University)*, Kristen Shinohara *(Rochester Institute of Technology)*, Kate Sonka *(Teach Access)*, Rolando Mendez *(Teach Access)*
- **Visual Data Science with Blockly-DS** 1892
  Luiz Barboza *(University of Memphis)*, Rafael Ferreira Mello *(Federal Rural University of Pernambuco)*, Erico Souza Teixeira *(CESAR)*, Andrew M. Olney *(University of Memphis)*
- **Autograding Python Code with the Pedal Framework: Feedback Beyond Unit Tests** 1893
  Austin Cory Bart *(Computer Science, University of Delaware)*, Luke Gusukuma *(Virginia Commonwealth University)*
- **From Informal Ed to the University – Hands-on Use of the BBC Micro:bit in CS1 Courses Post-COVID** 1893
  Albert Chan *(Fayetteville State University)*, Tieming Geng *(Fayetteville State University)*, Joseph Kabbes *(Fayetteville State University)*, Mingxian Jin *(Fayetteville State University)*, Longfei Wu *(Fayetteville State University)*
- **Foundational Tools for Coaching Data Storytelling** 1894
  Lujie Karen Chen *(University of Maryland, Baltimore County)*, Jiaqi Gong *(The University of Alabama)*, Louise Yarnall *(SRI International)*
- **Decompose Graphics to Compose Programs in Python with PyTamaro** 1894
  Luca Chiodini *(Università della Svizzera italiana)*, Matthias Hauswirth *(Università della Svizzera italiana)*
- **Effective DEI Committees: Sharing and Refining Key Practices** 1895
  Erika Dawson Head *(University of Massachusetts Amherst)*, Colleen M. Lewis *(University of Illinois Urbana-Champaign)*, Luther A. Tychonievich *(University of Illinois Urbana-Champaign)*
- **Research Expansion and Infrastructure Investment for Capacity Building** 1895
  Sonal Dekhane *(National Science Foundation)*, Subrata Acharya *(National Science Foundation)*
- **Develop a Competency-Based Curriculum that Purposefully Integrates Computing Skills, Cross-Disciplinary Skills, and Dispositions** 1896
  Marisa E. Exter *(Purdue University)*, Deepti Tagare *(Purdue University)*, Mihaela Sabin *(University of New Hampshire)*
- **Student Mission Control: Integrating Space Data Exploration into Data and Computer Science Education** 1896
  Dan Garcia *(University of California, Berkeley)*, Josh Hug *(University of California, Berkeley)*, Anirudhan Badrinath *(Stanford University)*, Michael Ball *(University of California, Berkeley)*, Lauren Mock *(University of California, Berkeley)*
- **Igniting Curiosity with BJC Sparks: A Transformative Curriculum for Middle and High School Computer Science** 1897
  Daniel Garcia *(University of California, Berkeley)*, Mary Fries *(Education Development Center)*, Michael Ball *(University of California, Berkeley)*, Lauren Mock *(University of California, Berkeley)*
- **A Quantum Abacus for Teaching Quantum Algorithms** 1897
  Dan-Adrian German *(Indiana University)*, Marcelo Pias *(FURG - Federal University of Rio Grande)*, Qiao Xiang *(Xiamen University)*

- **AI Enhanced Learning: Powering Curated Videos with Generative Intelligence** 1898
  Ananda Gunawardena *(Rutgers University, Princeton University)*, Naina Chaturvedi *(Rutgers University)*
- **Creating an On-ramp to Programming for Arts and Humanities Students with Teaspoon Languages and Custom Block Languages** 1898
  Mark Guzdial *(University of Michigan)*,
- **Project-based Activities to Introduce Hardware in a Software-focused Course** 1899
  Lama Hamandi *(Northeastern University)*,
  Mark Leslie Miller, Ph.D. *(Northeastern University and Learningtech.org)*,
  Shivakumar Mathapathi *(Northeastern University)*
- **Generative AI in Computer Science Education** 1899
  Orit Hazzan *(Technion - Israel Institute of Technology)*,
  Yael Erez *(Braude College of Engineering, Technion - Israel Institute of Technology)*
- **PLCC: A Tool Set for Teaching Programming Languages Courses** 1900
  Stoney Jackson *(Western New England University)*,
  James Heliotis *(Rochester Institute of Technology (Emeritus))*,
  Timothy Fossum *(SUNY College at Potsdam (Emeritus))*
- **Transform Your Computer Science Course with Specifications Grading** 1900
  David L. Largent *(Ball State University)*, Christian Roberson *(Florida Southern College)*,
  Linda F. Wilson *(Texas Lutheran University)*,
  Manuel A. Pérez-Quiñones *(University of North Carolina, Charlotte)*
- **Using Large Language Models for Teaching Computing** 1901
  Juho Leinonen *(School of Computer Science, The University of Auckland)*,
  Stephen MacNeil *(Computing and Information Sciences, Temple University)*,
  Paul Denny *(School of Computer Science, The University of Auckland)*,
  Arto Hellas *(Department of Computer Science, Aalto University)*
- **3D Java: Using Physical Objects to Make Abstract Concepts Concrete** 1901
  Colleen M. Lewis *(University of Illinois at Urbana-Champaign)*
- **Teaching with AI (GPT)** 1902
  Rongxin Liu *(Harvard University)*, Carter Zenke *(Harvard University)*,
  Doug Lloyd *(Harvard University)*, David J. Malan *(Harvard University)*
- **Exploring Java Programming after Java Version 8** 1902
  Shawn Lupoli *(Computer Science, Texas A&M University)*, Heather Stephens *(Oracle)*
- **Providing Students with Standardized, Cloud-Based Programming Environments at Term's Start (for Free)** 1903
  David J. Malan *(Harvard University)*, Rongxin Liu *(Harvard University)*,
  Carter Zenke *(Harvard University)*, Doug Lloyd *(Harvard University)*
- **Moving Beyond Game Design: Building Teacher Confidence with Scientific Modeling by Transitioning from Game Design to EcoSystem Programming** 1904
  Corrina Perrone Smith *(AgentSheets, Inc.)*, Nadia Repenning *(AgentSheets, Inc.)*
- **Ninth SPLICE Workshop on Technology and Data Infrastructure for CS Education Research** 1904
  Cliff Shaffer *(Virginia Tech)*, Peter Brusilovsky *(University of Pittsburgh)*,
  Ken Koedinger *(Carnegie Mellon University)*, Thomas Price *(North Carolina State University)*,
  Tiffany Barnes *(North Carolina State University)*, Behrooz Mostafavi *(North Carolina State University)*
- **Autograding Java Assignments in Gradescope with Jacquard** 1905
  Ellen Spertus *(Northeastern University)*, Priscilla Jiménez *(Grinnell College)*
- **Machine Learning on the Move: Teaching ML Kit for Firebase in a Mobile Apps Course** 1905
  Todd S. Sproull *(Washington University in St. Louis)*, Doug Shook *(Washington University in St. Louis)*
- **Integrating Parallel and Distributed Computing in Early Computing Classes** 1906
  Alan Sussman *(University of Maryland)*, Sushil Prasad *(University of Texas at San Antonio)*,
  Charles Weems *(University of Massachusetts)*, Sheikh Ghafoor *(Tennessee Tech University)*,
  Ramachandran Vaidyanathan *(Louisiana State University)*

SIGCSE TS 2024 Birds of a Feather

- **Engaging Rural Populations in Computer Science** 1907
  Nathan H. Bean *(Kansas State University)*, Russell Feldhausen *(Kansas State University)*,
  Joshua L. Weese *(Kansas State University)*, Michelle Friend *(University of Nebraska Omaha)*
- **Connecting the Computer Science Teachers Association (CSTA) Western Chapters** 1907
  Lauren Bricker *(University of Washington)*, Randy Macdonald *(Corvallis High School)*,
  Sean Glantz *(CSTA/Bend Senior High School)*
- **Primarily Undergraduate Institution Faculty** 1908
  Victoria Dean *(Olin College of Engineering)*, Lynn Kirabo *(Harvey Mudd College)*,
  Sophia Krause-Levy *(University of San Diego)*, Cynthia Taylor *(Oberlin College)*
- **Computing as a University Graduation Requirement** 1908
  Zachary Dodds *(Harvey Mudd College)*, Yuan Garcia *(Harvey Mudd College)*,
  Vidushi Ojha *(University of Illinois)*, Mark Guzdial *(University of Michigan)*,
  Tamara Nelson-Fromm *(University of Michigan)*, Valerie Barr *(Bard College)*,
  Stephanos Matsumoto *(Olin College)*
- **Experiences With Computer-Based Testing (CBT)** 1909
  Armando Fox *(University of California, Berkeley)*, Dan Garcia *(University of California, Berkeley)*,
  Cinda Heeren *(University of British Columbia)*, Firas Moosvi *(University of British Columbia)*,
  Mariana Silva *(University of Illinois, Urbana-Champaign)*,
  Matthew West *(University of Illinois, Urbana-Champaign)*,
  Craig Zilles *(University of Illinois, Urbana-Champaign)*
- **K-12 CS Teacher Education: What are We Teaching and to Whom? How do We know We are Moving the Needle?** 1909
  Charity Freeman *(Computer Science Teachers Association)*,
  Todd Lash *(University of Illinois, Urbana-Champaign)*
- **Creating University CS Teacher Preparation Programs** 1910
  Michelle Friend *(University of Nebraska Omaha)*,
  Jennifer Rosato *(Northern Lights Collaborative for Computing Education)*,
  Nathan Bean *(Kansas State University)*, Russell Feldhausen *(Kansas State University)*,
  Josh Weese *(Kansas State University)*
- **Integrating Computer Science in Elementary Education** 1910
  Lisa Garbrecht *(The University of Texas at Austin)*, Stephanie Baker *(The University of Texas at Austin)*,
  Zhuoying Wang *(The University of Texas at Austin)*
- **Teaching Track Faculty in Computer Science** 1911
  Olga Glebova *(University of Connecticut)*, Chris Gregg *(Stanford University)*,
  Melinda McDaniel *(Georgia Institute of Technology)*, Laney Strange *(Northeastern University)*
- **POGIL in Computer Science for Beginners and Experts** 1911
  Olga Glebova *(University of Connecticut)*, Helen H. Hu *(Westminster University)*,
  Kendra Walther *(University of Southern California)*
- **Humanitarian Open Source in the Classroom** 1912
  Gregory Hislop *(Drexel University)*, Heidi Ellis *(Western New England University)*,
  Darci Burdge *(Nassau Community College)*, Cameron Macdonell *(MacEwan University)*
- **Equity and Inclusion Considerations in CS Education for Students Living with Mental Health and Medical Conditions** 1912
  Maya Israel *(University of Florida)*, Catherine Law *(Oregon State University)*,
  Christian Murphy *(Swarthmore College)*
- **Two-Year Program Conversations: Supporting a Diversity of Students, Articulation Pathways, and More** 1913
  William Kerney *(Clovis Community College)*, Laura Malave *(St. Petersburg Community College)*,
  William Pulling *(Fanshawe College)*, Cara Tang *(Portland Community College)*
- **Finding Neurodivergent Community in Computing Education** 1913
  Mara Kirdani-Ryan *(University of Washington, Seattle)*, Omar Ibrahim *(University of Illinois Chicago)*

- **Computing, Education, and Capitalism** 1914
  Amy J. Ko *(University of Washington)*, Francisco Castro *(New York University)*,
  Aakash Gautam *(University of Pittsburgh)*, Anne Drew Hu *(Michigan State Univesity)*,
  Sara Kingsley *(Carnegie Mellon University)*, Michael Lachney *(Michigan State University)*,
  Aman Yadav *(Michigan State University)*
- **Community-based Service Learning: Best Practices in Software Projects with Community Partners** 1914
  Stan Kurkovsky *(Central Connecticut State University)*, Chad Williams *(Central Connecticut State University)*,
  Mikey Goldweber *(Denison University)*, Nathan Sommer *(Xavier University)*
- **Disability and Accessibility in Computer Science Education** 1915
  Richard E. Ladner *(University of Washington)*, Brianna Blaser *(University of Washington)*,
  Andreas Stefik *(University of Nevada, Las Vegas)*, Amy J. Ko *(University of Washington)*,
  Raja Kushalnagar *(Gallaudet University)*
- **BOF: Grading for Equity in Computer Science Courses** 1915
  David L. Largent *(Ball State University)*, Manuel A. Pérez-Quiñones *(University of North Carolina, Charlotte)*,
  Christian Roberson *(Florida Southern College)*, Linda F. Wilson *(Texas Lutheran University)*,
  Adrienne Decker *(University at Buffalo)*, Stephen H. Edwards *(Virginia Tech)*
- **Birds of a Feather Who'd Like to Share Software Together: Teaching Tools that Improve Efficiency and Outcomes** 1916
  Doug Lloyd *(Harvard University)*, Carter Zenke *(Harvard University)*, David J. Malan *(Harvard University)*
- **Discussing the Changing Landscape of Generative AI in Computing Education** 1916
  Stephen MacNeil *(Temple University)*, Juho Leinonen *(Aalto University)*,
  Paul Denny *(The University of Auckland)*,
  Natalie Kiesler *(DIPF Leibniz Institute for Research and Information in Education)*,
  Arto Hellas *(Aalto University)*, James Prather *(Abilene Christian University)*,
  Brett A. Becker *(University College Dublin)*, Michel Wermelinger *(The Open University)*,
  Karen Reid *(University of Toronto)*
- **Can We Build an Excellent Undergraduate TA Program? Fostering Collaboration and Developing Best Practices** 1917
  Melinda McDaniel *(Georgia Institute of Technology)*, Fisayo Omojokun *(Georgia Institute of Technology)*,
  Mary Hudachek-Buswell *(Georgia Institute of Technology)*
- **Building Community for Graduate Students in CS Education Research** 1917
  Tamara Nelson-Fromm *(University of Michigan)*, Grace Barkhuff *(Georgia Institute of Technology)*,
  Jayne Everson *(University of Washington)*, Morgan M. Fong *(University of Illinois, Urbana-Champaign)*,
  Elijah Rivera *(Brown University)*
- **Sourcing Projects for CS Capstones: Challenges and Strategies** 1918
  Olufisayo Omojokun *(Georgia Institute of Technology)*,
  KellyAnn Fitzpatrick *(Georgia Institute of Technology)*
- **Minority Serving Institution Meet-Up** 1918
  Jody Paul *(Computer Sciences, Metropolitan State University of Denver)*
- **Cultivating and Celebrating LGBTQ+ Community in Computing Education** 1919
  Joslenne Peña *(Macalester College)*, F. Megumi Kivuva *(University of Washington)*,
  Mara Kirdani-Ryan *(University of Washington)*, Francisco Enrique Vicente Castro *(New York University)*,
  Amy J. Ko *(University of Washington)*
- **BOF: Hispanics in Computing** 1919
  Brianna Posadas *(Virginia Tech)*, Oscar Veliz *(Northeastern University)*,
  Patricia Ordóñez *(University of Maryland, Baltimore County)*,
  Manuel A. Pérez-Quiñones *(University of North Carolina, Charlotte)*
- **Topics for an Introductory Data Science Course** 1920
  Michael A. Posner *(Villanova University)*, April Kerby-Helm *(Winona State University)*
- **A Town Meeting: SIGCSE Committee on Expanding the Women-in-Computing Community** 1920
  Wendy Powley *(Queen's University)*, Gloria Townsend *(DePauw University)*

- **Teaching Ethics In CS Programs - Questions, Models, Resources, Assessments** 1921
  Vance Ricks *(Northeastern University)*, Meica Magnani *(Northeastern University)*,
  Casey Fiesler *(University of Colorado-Boulder)*, Grace Barkhuff *(Georgia Institute of Technology)*,
  Stacy Doore *(Colby College)*, Alexi Brooks *(University of Wisconsin-Stout)*,
  Michael Zimmer *(Marquette University)*
- **Directing Undergraduate Studies: Sharing Practices and Problems** 1921
  Bill Siever *(Washington University in St. Louis)*
- **Platform or Foundational Concept? Approaches to Teaching Modern Web Development** .. 1922
  Michael C. Stewart *(James Madison University)*, Michael S. Kirkpatrick *(James Madison University)*
- **Mentoring, AI, and the End of Affirmative Action: Connecting with SIGCSE Reads** 1922
  Nanette Veilleux *(Simmons University)*, Rebecca Bates *(Minnesota State University, Mankato)*,
  Judy Goldsmith *(University of Kentucky)*, Valerie Summet *(Rollins College)*

SIGCSE TS 2024 Demonstrations

- **AntiCopyPaster: An Open-Source Ecosystem for Just-in-time Code Duplicates Extraction** .. 1923
  Eman Abdullah AlOmar *(Stevens Institute of Technology)*,
  Mohamed Wiem Mkaouer *(University of Michigan-Flint)*
- **Embedding Executable Code in Slides for Introductory Programming: The Case of Interactive Code Playgrounds** 1923
  Lorenzo Angeli *(University of Trento)*, Luca De Menego *(Thema Optical)*,
  Maurizio Marchese *(University of Trento)*
- **Snap! 9 — Support for Teachers and Programming with Data** 1924
  Michael Ball *(UC Berkeley)*, Daniel Garcia *(UC Berkeley)*, Victoria Phelps *(UC Berkeley)*,
  Yuan Garcia *(Harvey Mudd College)*
- **UWCSE351 Virtual Memory Simulator** 1924
  Kunchen Dai *(University of Washington)*, Justin Hsia *(University of Washington)*
- **Google Colab for Teaching CS and ML** 1925
  Katlyn Edwards *(Colab, Google)*, Corrie Scalisi *(Colab, Google)*, Julianne DeMars-Smith *(Colab, Google)*,
  Key Lee *(Colab, Google)*
- **Algovision - An Algorithm Visualization Tool** 1925
  Luděk Kučera *(Faculty of Mathematics and Pysics, Charles University)*
- **Embodied Code: Creative Coding in Virtual Reality** 1926
  Ryan Lay *(University of California, San Diego)*, Rhea Bhutada *(University of California, San Diego)*,
  Alejandro Lobo *(University of California, San Diego)*, Robert Twomey *(University of Nebraska-Lincoln)*,
  Amy Eguchi *(University of California, San Diego)*, Ying Choon Wu *(University of California, San Diego)*
- **Providing Students with Standardized, Cloud-Based Programming Environments at Term's Start (for Free)** 1927
  Rongxin Liu *(Harvard University)*, Charlie Liu *(Yale University)*,
  Carter Zenke *(Harvard University)*, David J. Malan *(Harvard University)*
- **Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education** 1928
  Rongxin Liu *(Harvard University)*, Carter Zenke *(Harvard University)*,
  Charlie Liu *(Yale University)*, Andrew Holmes *(Harvard University)*, Patrick Thornton *(Harvard University)*,
  David J. Malan *(Harvard University)*
- **Bringing a Visual Memory Model to VS Code** 1928
  Matt Schwartz *(Kansas State University)*, Nathan H. Bean *(Kansas State University)*,
  Joshua L. Weese *(Kansas State University)*, Russell Feldhausen *(Kansas State University)*
- **Smart Greenhouse: Bridging Physical Computing, Plant Science, and Data Literacy** 1929
  Sheikh Ahmad Shah *(Boston College)*, Jaai Uday Phatak *(Boston College)*,
  Avneet Hira *(Boston College)*, Helen Zhang *(Boston College)*, Barnet *(Boston College)*
- **Social Media, Ethics, and Automation: Introducing a Free Interactive Online Textbook** 1929
  Kyle Thayer *(University of Washington)*, Susan Notess *(Durham University)*

- **Algot: A Visual, Hands-On Approach to Introductory Computer Science** 1929
  Sverrir Thorgeirsson *(ETH Zürich)*, Theo B. Weidmann *(ETH Zürich)*,
  Sara Hooshangi *(Virginia Tech)*
- **An Environment for Algorithmic Essays** 1930
  Michel Wermelinger *(The Open University)*, Michael Snowden *(The Open University)*
- **Let's Build a Traffic Light: Teaching Physical Computing to Children with EdBoard** 1930
  Ruhan Yang *(University of Colorado Boulder)*, Ellen Yi-Luen Do *(University of Colorado Boulder)*

SIGCSE TS 2024 Lightning Talks

- **How can We Leverage Static Analysis and Large Language Models to Engage Students in Software Quality Improvement** 1930
  Eman Abdullah AlOmar *(Stevens Institute of Technology)*,
  Mohamed Wiem Mkaouer *(University of Michigan-Flint)*
- **Using Natural Language Processing to Explore Instructional Change Strategies in Undergraduate Science Education Literature** 1930
  Emily Bolger *(Michigan State University)*, Marcos Caballero *(Michigan State University)*
- **Registered Reports: A New Way to Publish Papers** 1931
  Neil C. C. Brown *(King's College London)*
- **Moving forward with *LogicWriter Actual,* A Web App for Early Undergraduate Writing with Mathematical Logic** 1931
  Bruce W. Char *(Drexel University)*, Jeremy Johnson *(Drexel University)*,
  Steve Earth *(Drexel University)*
- **A Vision for the Next 15 Years of Computing Education** 1932
  Adrienne Decker *(University at Buffalo)*, Mark Allen Weiss *(Florida International University)*
- **Active Learning at Large-Scale: Using Video Tutorials to Learn by Teaching** 1932
  Pedro Guillermo Feijóo-García *(Georgia Institute of Technology)*, Nimisha Roy *(Georgia Institute of Technology)*
- **Quantifying the Effects of Advanced Placement Computer Science Principles on Participation and Diversity in AP CS** 1933
  Daniela Ganelin *(Stanford University)*, Thomas S. Dee *(Stanford University)*
- **Develop Innovative Virtual Reality Interaction Designs for the Learning of Computer Science Concepts and Theories** 1933
  Chunming Gao *(Central Washington University)*
- **A Comparative Study on Student and Faculty Perceptions of Online Computing Labs** 1934
  Yunfei Hou *(California State University, San Bernardino)*,
  Miranda McIntyre *(California State University, San Bernardino)*,
  Jesus Herrera *(California State University, San Bernardino)*, Joyce Fu *(University of California, Riverside)*,
  Hani Aldirawi *(California State University, San Bernardino)*,
- **Integrating Critical Analysis of Society and Technology into K-12 Computing Through Teacher Co-Design** 1934
  Anne Drew Hu *(Michigan State University)*, Aman Yadav *(Michigan State University)*
- **Scaling Responsible Computing Globally: Lessons from the US, Kenya, and India** 1935
  Crystal Lee *(Massachusetts Institute of Technology)*, Chao Mbogo *(Mozilla Foundation)*,
  Jibu Elias *(Mozilla Foundation)*, Joycelyn Streator *(Prairie View A&M University)*,
  Kathy Pham *(Harvard University)*, Ziyaad Bhorat *(University of Southern California)*,
  Steven Azeka *(Columbia University)*
- **Self-efficacy Interventions for CS1** 1935
  Alex Lishinski *(University of Tennessee, Knoxville)*, Hanhui Bao *(University of Tennessee, Knoxville)*,
  Joshua Rosenberg *(University of Tennessee, Knoxville)*
- **Creative Labs in a CS1 Course: Self-directed Labs Enhance Inclusivity in Computer Science Learning** 1936
  Melinda McDaniel *(Georgia Institute of Technology)*

- **Moms Can Be Computing Leaders, Too! Why We Need Community Computing Learning Centers Dsigned and Led by Mothers** 1936
  Patricia Ordóñez *(University of Maryland, Baltimore County)*
- **Data Analytics for Social Good: A Collaborative Fusion of Computer Science and Social Science** 1937
  Tina J. Ostrander *(Green River College)*, Tim Scharks *(Green River College)*, Kendrick Hang *(Green River College)*
- **A Cross-disciplinary Review of Introductory Undergraduate Data Science Course Content** 1937
  Michael A. Posner *(Villanova University)*, April Kerby-Helm *(Winona State University)*, Alana Unfried *(California State University, Monterey Bay)*, Douglas Whitaker *(Mount Saint Vincent University)*, Marjorie E. Bond *(The Pennsylvania State University)*, Leyla Batakci *(Elizabethtown College)*
- **Exposing the Hidden Curriculum with a First Year Computing Seminar** 1938
  Derek Riley *(Milwaukee School of Engineering)*, Katherine Panciera *(Milwaukee School of Engineering)*
- **Enabling Widespread Engagement in DS and AI: The Generation AI Curriculum Initiative for Community Colleges** 1938
  Rebecca Schroeder *(The University of Texas at San Antonio)*, Jianwei Niu *(The University of Texas at San Antonio)*, Ashwin Malshe *(The University of Texas at San Antonio)*, Sue Hum *(The University of Texas at San Antonio)*, Siobhan Flemming *(The University of Texas at San Antonio)*, Ian Thacker *(The University of Texas at San Antonio)*
- **Keeping Software Engineering Curriculum Relevant** 1939
  Simon Sultana *(Reedley College)*, James D. Kiper *(Miami University)*, Brent Auernheimer *(California State University, Fresno)*, Gursimran Singh Walia *(Augusta University)*
- **DEEILS: Data Ethics Embedded Interactive Learning System for Computer Science Students** 1939
  Ke Yang *(University of Texas at San Antonio)*
- **The Role of Probing and Clarifying Questions for Teaching Fellows in Computer Science: Guiding Student Growth** 1940
  Yuliia Zhukovets *(Harvard University)*, Carter Zenke *(Harvard University)*, David J. Malan *(Harvard University)*

SIGCSE TS 2024 Nifty Assignments

- **Alphabear Partial Solver** 1941
  Ryan Dougherty *(United States Military Academy)*
- **Exploring the Scurry of Squirrels in Central Park** 1941
  Rachel Lyn-Salmon Frisbie *(Michigan State University)*, Devin Silvia *(Michigan State University)*, Marcos D. Caballero *(Michigan State University)*, Rachel Roca *(Michigan State University)*, Amanda Bowerman *(Michigan State University)*, Krithi Sachithanand *(Michigan State University)*
- **Simulating Election Votes** 1942
  Aleksandr Popov *(National Research University Higher School of Economics)*
- **Infinity War** 1942
  Yashas Ravi *(Rutgers University - New Brunswick)*, Ana Paula Centeno *(Rutgers University - New Brunswick)*
- **The Fingerprint Assignment: An Interdisciplinary Assessment for CS I Education** 1943
  Christian Servin *(El Paso Community College)*, Ivan Alonso *(El Paso Community College)*, Emiliano Garcia *(El Paso Community College)*
- **Modular Virtual 3D Cities Assignment** 1943
  Christopher John Tralie *(Ursinus College)*

Author Index 1944

SIGCSE TS 2024 Symposium Organization

Symposium Co-Chairs: Ben Stephenson (University of Calgary)
Jeffrey A. Stone (Penn State University)

Program Co-Chairs: Lina Battestilli (North Carolina State University)
Samuel A. Rebelsky (Grinnell College)
Libby Shoop (Macalester College)

Hybrid Experience Co-Chairs: Delaram Yazdansepas (Loyola Marymount University)
Oluwakemi Ola (University of British Columbia)
Kristin Stephens-Martinez (Duke University)

Program Committees

ACM Student Research Competition: Mohammed Seyam (Virginia Tech)
Timothy Yue (University of Texas at San Antonio)

Birds of a Feather: Kevin Lin (University of Washington, Seattle)
Adam Blank (Caltech)

Demos and Lightning Talks: Eric Fouh (University of Pennsylvania)
Lisa Lacher (University of Houston-Clear Lake)

Nifty Assignments: Dave Reed (Creighton University)
Michael Guerzhoy (University of Toronto)
Nick Parlante (Stanford University)
Julie Zelenski (Stanford University)

Panels and Special Sessions: S. Monisha Pulimood (The College of New Jersey)
Brian Harrington (University of Toronto Scarborough)

Posters: Meghan Allen (University of British Columbia)
Ashish Aggarwal (University of Florida)

Workshops: Mia Minnes (UC San Diego)
Bedour Alshaigy (Uppsala University)
James Prather (Abilene Christian University)

Steering Committee

Chair: Sarah Heckman (North Carolina State University)

Secretary: Steve Wolfman (University of British Columbia)

Members: Leen-Kiat Soh (University of Nebraska-Lincoln)
Kurt Eiselt (University of California, Davis)

SIGCSE Board Liaison: Adrienne Decker (University at Buffalo)

Organizing Committees

Affiliated Events:	Matthew Hertz (University at Buffalo) Abdussalam Alawini (University of Illinois at Urbana Champaign)
Community College Liaison:	Nadia Karichev (El Paso Community College)
Evaluations:	Amanpreet Kapoor (University of Florida) Margaret Ellis (Virginia Tech)
Global Liaisons:	Diane Cukierman (Simon Fraser University) Ethel Tshukudu (University of Glasgow)
K-12 Liaisons:	Mohsen Dorodchi (University of North Carolina Charlotte) David Czechowski
Kids' Camp:	Karen Arlien (Bismarck State College) Eleni Zamagias (St. Joseph's University, New York)
Local Arrangements:	Dustin Bessette (Mt. Hood Community College) Lucas Cordova (Willamette University)
Minority Serving Institution Liaisons:	Jody Paul (Metropolitan State University of Denver) Debzani Deb (Winston-Salem State University)
Publications:	Lasanthi Gamage (Webster University) TJ OConnor (Florida Institute of Technology)
Publicity and Social Media:	Rasika Bhalerao (Northeastern University) Christian Yongwhan Lim (Columbia University)
Puzzles and Games:	Robbie Hott (University of Virginia) Melinda McDaniel (Georgia Institute of Technology)
Registration:	Briana B. Morrison (University of Virginia) Albert Chan (Fayetteville State University) Craig S. Miller (DePaul University)
Student Volunteers and Activities:	Victoria Eisele (Front Range Community College) Benjamin Say (Metropolitan Community College) Joslenne Peña (Macalester College)
Sponsor/Exhibitor Liaisons:	Jian Zhang (Texas Woman's University) Kristine Nagel (Georgia Institute of Technology)
Treasurer:	Deepti Joshi (The Citadel)
Universal Design Chairs:	Yasmine Elglaly (Western Washington University) Earl Huff Jr (The University of Texas at Austin)

Universal Design Committee: Briana Bettin (Michigan Technological University)
Siobahn Day Grady (North Carolina Central University)
Becky Grasser (Lakeland Community College)
Natalie Kiesler (DIPF | Leibniz Institute for Research and Information in Education)

Web and Data: Adam Blank (Caltech)
Jérémie Lumbroso (University of Pennsylvania)

Associate Program Chairs: Ildar Akhmetov (Northeastern University)
Bita Akram (North Carolina State University)
Jennifer Albert (The Citadel)
Andrew Allen (Georgia Southern University)
Julio Bahamon (UNC Charlotte)
Matthew Barr (University of Glasgow)
Brett Becker (University College Dublin)
Marc Berges (Friedrich-Alexander Universität Erlangen-Nürnberg)
Bradley Beth (Vermont State University)
Jeremiah Blanchard (University of Florida)
Dennis Bouvier (United States Air Force Academy)
Rachel Bradley (University of Nebraska-Lincoln)
Dennis Brylow (Marquette University)
Antonio Bucchiarone (FBK-IRST)
Kevin Buffardi (California State University - Chico)
Zack Butler (Rochester Institute of Technology)
Yu Cai (Michigan Technological University)
Veronica Catete (North Carolina State University)
Albert Chan (Fayetteville State University)
Daniel Cliburn (University of the Pacific)
Steve Cooper (University of Nebraska - Lincoln)
Diana Cukierman (Simon Fraser University)
Kathryn Cunningham (University of Illinois Urbana-Champaign)
Matthew Curinga (Adelphi University)
Pam Cutter (Kalamazoo College)
James H. Davenport (University of Bath)
Adrienne Decker (University at Buffalo)
Catherine Delcourt (Wellesley College)
Leigh Ann Delyser (CSforALL)
Paul Denny (The University of Auckland)
Laura Dillon (Michigan State University)

Associate Program Chairs (continued):
Anca Doloc-Mihu (Georgia Gwinnett College)
Mohsen Dorodchi (University of North Carolina Charlotte)
Maureen Doyle (Northern Kentucky University)
Bob Edmison (Virginia Tech)
Barbara Ericson (University of Michigan)
Matthew Forshaw (Newcastle University)
Ge Gao (North Carolina State University)
Dan Garcia (University of California, Berkeley)
Rita Garcia (Victoria University of Wellington)
Sara Gopalkrishna (George Mason University)
Chris Gregg (Stanford University)
Mark Guzdial (University of Michigan)
Steve Hadfield (US Air Force Academy)
Sarah Heckman (North Carolina State University)
Cecily Heiner (Oregon Institute of Technology)
Geoffrey Herman (University of Illinois at Urbana-Champaign)
Michael Hilton (Carnegie Mellon University)
Sara Hooshangi (Virginia Tech)
Helen Hu (Westminster University)
Cruz Izu (The University of Adelaide)
Shuyin Jiao (North Carolina State University)
Yerika Jimenez (Duke University)
Swaroop Joshi (BITS Pilani, KK Birla Goa Campus, India)
David Joyner (Georgia Institute of Technology)
Amanpreet Kapoor (University of Florida)
Ayaan Kazerouni (California Polytechnic State University)
Carsten Kleiner (University of Applied Science and Arts Hannover)
David Largent (Ball State University)
Juho Leinonen (Aalto University)
David Levine (Saint Bonaventure University)
J. Jenny Li (NSF/Kean)
Collin Lynch (North Carolina State University)
Francesco Maiorana (University of Urbino)
Daniela Marghitu (Auburn University)
Maíra Marques Samary (Boston College)
William Mason (University of British Columbia)
Suzanne Matthews (United States Military Academy)
Bruce Maxim (University of Michigan)
Lester McCann (University of Arizona)
Brian McSkimming (University of Oklahoma)
Laurence D. Merkle (Air Force Institute of Technology)
Briana Morrison (University of Virginia)
Kris Nagel (Georgia Institute of Technology)

Associate Program Chairs (continued):

Kristine Nagel (Georgia Institute of Technology)
Narges Norouzi (UC Berkeley)
Michael Nowak (University of Illinois at Urbana-Champaign)
Jennifer Parham-Mocello (Oregon State University)
Miranda Parker (San Diego State University)
James Paterson (Glasgow Caledonian University)
Raymond Pettit (University of Virginia)
Brian Plancher (Barnard College, Columbia University)
Wendy Powley (Queen's University)
Upsorn Praphamontripong (University of Virginia)
Thomas Price (North Carolina State University)
Sagar Raina (Mount Saint Mary College)
Preeti Raman (Toronto Metropolitan University)
Jake Renzella (University of New South Wales, Sydney)
Susan Rodger (Duke University)
Jean Rosa (Interactive Technologies Institute)
Guido Rößling (TU Darmstadt)
Mark Sherriff (University of Virginia)
Swapneel Sheth (University of Pennsylvania)
Leen-Kiat Soh (University of Nebraska- Lincoln)
Gerald Soosairaj (University of California San Diego)
Rebecca Strachan (Northumbria University)
Kalpathi Subramanian (The University of North Carolina at Charlotte)
Cheryl Swanier (University of Massachusetts - Amherst)
Cynthia Taylor (Oberlin College)
Ben Tribelhorn (University of Portland)
Luther Tychonievich (University of Illinois at Urbana-Champaign)
Frank Vahid (UC Riverside)
Tammy Vandegrift (University of Portland)
J. Ángel Velázquez-Iturbide (Universidad Rey Juan Carlos)
Henry Walker (Grinnell College)
Charles Wallace (Michigan Technological University)
Josh Weese (Kansas State University)
Angela Zavaleta Bernuy (University of Toronto)

SIGCSE TS 2024 Reviewers

Fadi Abuamara (Shenandoah University)
Hezekiah Seun Adewinbi (Kent State University)
Arav Agarwal (Carnegie Mellon University)
Nimisha Agarwal (Indian Institute of Technology Kanpur)
Friday Joseph Agbo (Willamette University)
Marzieh Ahmadzadeh (York University)
Mahir Akgun (The Pennsylvania State University)
Ildar Akhmetov (Northeastern University)
Saima Akhtar (Barnard College, Columbia University)
Bita Akram (North Carolina State University)
Nkechi Akwari (University of Cincinnati)
Shah Jamal Alam (Habib University)
Abdussalam Alawini (University of Illinois at Urbana-Champaign)
Ibrahim Albluwi (Princeton University)
Roselie Alday (Lyceum of the Philippines University - Batangas)
Peter Aldous (Utah Valley University)
Mona Ali (Northeastern university)
Reem Ali (University of California Riverside)
Amin Alipour (University of Houston)
Vicki Allan (Utah State Universtiy)
Atheer Almogbil (Johns Hopkins University)
Vicki Almstrum (Computing Educators Oral History Project)
Eman Alomar (SIT)
Chris Alvin (Furman University)
Heather Amthauer (College of Saint Benedict and Saint John's University)
Marshall An (Carnegie Mellon University)
Asm Iftekhar Anam (University of Wisconsin - Green Bay)
Daniel Anderson (Carnegie Mellon University)
Emma Anderson (Northumbria University)
Neil Anderson (Queen's University Belfast)
Scott Anderson (Wellesley College)
Ruth Anderson (University of Washington)
Wilkerson L. Andrade (Federal University of Campina Grande)
Oana Andrei (University of Glasgow)
Lorenzo Angeli (University of Trento)
Imad Antonios (Southern Connecticut State University)
John Anvik (University of Lethbridge)
Ajayi Anwansedo (Southern University and A & M)
Blancaflor Arada (University of the Philippines Open University)
Hamzah Arishi (University of Adelaide)
Chulakorn Aritajati (The Pennsylvania State University)
Ken Arnold (Calvin University)
John Aromando (University of Delaware)
Rosa Arriaga (Georgia Institute of Technology)
Brent Auernheimer (California State University, Fresno)
Idowu Awoyemi (The University of Alabama)
Ilknur Aydin (Farmingdale State college)
Victor Ayodele (Northumbria University)
Noushin Azami (Texas State University)
John Bacher (North Carolina State University)
Mario Badr (University of Toronto)
Donald Bagert (Benedictine College)
Tobias Bahr (University of Stuttgart)
Gina Bai (North Carolina State University)
Prof Hema Banati (Dyal Singh College, University of Delhi)
Rabin Banjade (University of Memphis)
Hanhui Bao (The University of Tennessee, Knoxville)
Cathy Bareiss (Bethel University)
William Bares (Whitman College)
Lecia Barker (University of Colorado Boulder)
Grace Barkhuff (Georgia Institute of Technology)
Joao Paulo Barros (Instituto Politecnico de Beja)
Phillip Barry (University of Minnesota)
Austin Bart (University of Delaware)
John Bartucz (University of Minnesota)
Arinjoy Basak (Virginia Tech)
Prateek Basavaraj (American Association of State Colleges and Universities)
Nada Basit (University of Virginia)
María Cecilia Bastarrica (Universidad de Chile)
Daniel Bauer (Columbia University)
Michael Bauer (Leeward Community College)
Gillian Bausch (SUNY Albany)
Nathan Bean (Kansas State University)
Jonathan Beaumont (University of Michigan)
Mattox Beckman (University of Illinois at Urbana-Champaign)
Patrice Belleville (The University of British Columbia)
Jamie Benario (Google)
Nadine Bergner (RWTH Aachen)
Maverick Berner (Marquette University)
Andrew Berns (University of Northern Iowa)
Ivona Bezakova (Rochester Institute of Technology)
Adeel Bhutta (Northeastern University)
William Billingsley (University of New England)
Anastasiia Birillo (JetBrains Research)
Jean Blair (United States Military Academy)
Brianna Blaser (University of Washington)
Aaron Bloomfield (University of Virginia)
Christopher Bogart (Carnegie Mellon University)

Christopher Bohn (University of Nebraska-Lincoln)
Rodrigo Borela (Georgia Institute of Technology)
Thomas Borrelli (Rochester Institute of Technology)
Chris Bourke (University of Nebraska-Lincoln)
Lonnie Bowe (Concord University)
Brook Bowers (University of Georgia)
Rachel Bradley (University of Nebraska-Lincoln)
Steven Bradley (Durham University)
Stefan Brass (University of Halle-Wittenberg)
Grant Braught (Dickinson College)
Katherine Breeden (Harvey Mudd College)
Johanna Brewer (Smith College)
Lauren Bricker (University of Washington)
Janet Brock (North Carolina State University)
Allan Brockenbrough (Salem State University)
Andrej Brodnik (University of Ljubljana)
Alexander Brooks (University of Wisconsin-Stout)
Julian Brooks (University of Leeds)
Neil Brown (King's College London)
Noelle Brown (University of Utah)
Laura E. Brown (Michigan Technological University)
Matthew Brown (Indiana University)
Jonathan Browning (Queen's University Belfast)
Nathan Brunelle (University of Virginia)
Paul Bruno (University of Illinois at Urbana-Champaign)
Sarita Bruschi (University of Sao Paulo)
Philip Buffum (Georgetown University)
David Bunde (Knox College)
Alyxander Burns (Mount Holyoke College)
Eden Burton (Seneca College of Applied Arts and Technology)
Yu Cai (Michigan Technological University)
Mary Elaine Califf (Illinois State University)
Jonathan Calver (University of Toronto)
Jennifer Campbell (University of Toronto)
Oladele Campbell (Niger State Polytechnic)
Manuel I. Capel Tuñón (none)
Alexander Card (North Carolina State University)
Preston Carman (Walla Walla University)
Virginia Carneiro de Paula (Palm Beach State College)
Leandro Carvalho (Institute of Computing - Universidade Federal do Amazonas)
Sarah Castle (University of Idaho)
Francisco Enrique Vicente Castro (New York University)
Jonathan Cazalas (Florida Southern College)
Martine Ceberio (University of Texas at El Paso)
Ana Paula Centeno (Rutgers University)
Younghun Chae (Kent State University)
Henry Chai (Carnegie Mellon University)
Subhajit Chakrabarty (LSU Shreveport)
Geoffrey Challen (University of Illinois)
Mohit Chandarana (Codio)
Aishwarya Chandrasekaran (University of Delaware)
Santosh Chandrasekhar (University of California, Merced)
Hung-Fu Chang (University of Indianapolis)
Alex Chao (Duke University)
Jeevan Chapagain (University of Memphis)
Bruce Char (Drexel University)
Prosenjit Chatterjee (Southern Utah University)
Mahesh Chaudhari (University of San Francisco)
Divya Chaudhary (Northeastern.edu)
Ajit Chavan (Cornell College)
Victoria Chávez (Northwestern University)
Binglin Chen (University of Illinois Urbana-Champaign)
Huanyi Chen (University of Waterloo)
Jingshu Chen (Oakland university)
Lujie Chen (University of Maryland, Baltimore County)
Xi Chen (Utah Valley University)
Ying Chen (University of Illinois system -IWERC)
Karen Cheng (CSforALL)
Qiong Cheng (UNC Charlotte)
Anjum Chida (University of Texas at Dallas)
Meng-Chieh Chiu (University of Massachusetts Amherst)
Ming-Hsing Chiu (Dillard University)
Sue Chng (University of Sydney)
Apoorve Chokshi (University of Calgary)
Venkatesh Choppella (International Inst. of Info. Technology Hyderabad)
Tahiya Chowdhury (Davis Institute for AI)
Beau Christ (Wofford College)
Vincent Cicirello (Stockton University)
John Cigas (Park University)
Christopher Cischke (Michigan Technological University)
Peter Clarke (Florida International University)
Jody Clarke-Midura (Utah State University)
Alison Clear (EIT)
Alexis Cobo (University of Florida)
Merijke Coenraad (Digital Promise)
Joel Coffman (United States Air Force Academy)
Shaanan Cohney (University of Pennsylvania)
Harold Connamacher (Case Western Reserve University)
Lucas Cordova (Western Oregon University)
Lino Coria (Northeastern University)
Jonathan Corley (University of West Georgia)
Rafael Corsi Ferrao (Insper)
Julian Costa (Pace University)
Michael Cotterell (University of Georgia)
Noah Cowit (University of Colorado Boulder)
Eugene Cox (University of Illinois at Urbana-Champaign)
Jennifer Coy (Ball State University)
Michelle Craig (University of Toronto)
Kathleen Creel (Northeastern University)

Ryan Crosby (Durham University)
Laura Cruz Castro (University of Florida)
Amy Csizmar Dalal (Carleton College)
Margaret Cullen (Nelson Mandela University Business School)
Lisa Cullington (National University)
Matthew Curinga (Adelphi University)
Pam Cutter (Kalamazoo College)
Lawrence D'Antonio (Ramapo College)
Mai Dahshan (University of North Florida)
Chaianun Damrongrat (National Electronics and Computer Technology Center (NECTEC))
Arghya Kusum Das (University of Alaska Fairbanks)
Udayan Das (Saint Mary's College of California)
Venu Dasigi (Bowling Green State University)
Janet Davis (Whitman College)
Rahmad Dawood (Universitas Syiah Kuala)
Julia Dean (CU Boulder)
Victoria Dean (Olin College of Engineering)
Anita Debarlaben (Computer Science Teacher Association (CSTA))
Christophe Debruyne (Université de Liège)
Daniel DeFreez (Southern Oregon University)
Alfred Degennaro (SC Governor's School for Science and Mathematics)
Nasrin Dehbozorgi (Kennesaw State University)
Elise Deitrick (Codio)
Sonal Dekhane (Georgia Gwinnett College)
Leigh Ann DeLyser (CSforALL)
Giorgio Delzanno (DIBRIS, Università di Genova)
Jill Denner (ETR Associates)
Benjamin Denzler (University of California, Riverside)
Dhruv Hemang Dhamani (University of North Carolina, Charlotte)
Diana Diaz Herrera (Clinical Assistant Professor)
Jaime Díaz-Arancibia (Universidad de La Frontera)
Ethan Dickey (Purdue University)
Zachary Dodds (Harvey Mudd College)
Emma Dodoo (University of Michigan)
Ignacio X. Domínguez (North Carolina State University)
Molly Domino (Virginia Tech)
Benoit Donnet (Université de Liège)
Stacy Doore (Colby College)
Niels Doorn (Open Universiteit)
Ryan Dougherty (United States Military Academy)
Rebecca Dovi (CodeVA)
Kelly Downey (University of California, Riverside)
Dylan Doyle (University of Colorado Boulder)
Geoffrey M. Draper (Brigham Young University Hawaii)
Hanxiang Du (Western Washington University)
Wendy Dubow (NCWIT)
Alex Duncan (Georgia Institute of Technology)
Rodrigo Duran (IFMS)
Kathleen Durant (Northeastern University)
Cyntrica Eaton (Norfolk State University)
Maria Ebling (United States Military Academy, West Point)
Bob Edmison (Virginia Tech)
Stephen Edwards (Virginia Tech)
Bobbie Eicher (Georgia Institute of Technology)
Jesse Eickholt (Central Michigan University)
Victoria Eisele (Front Range Community College, Larimer Campus)
Kurt Eiselt (University of California, Davis)
Laila El-Hamamsy (Ecole Polytechnique Fédérale de Lausanne (EPFL))
Heidi Ellis (Western New England University)
Margaret Ellis (Virginia Tech)
Daniel Ellsworth (Colorado College)
Maher Elshakankiri (University of Toronto)
Salma Emara (University of Toronto)
Chinny Emeka (University of Illinois at Urbana-Champaign)
Madeline Endres (University of Michigan)
Rutwa Engineer (University of Toronto)
Megan Englert (University of Colorado Boulder)
Rosanne English (Strathclyde university)
Yael Erez (Technion)
Mehmet Ergezer (Wentworth Institute of Technology)
Svana Esche (Technical University of Darmstadt)
John Estell (Ohio Northern University)
Matthew Evett (Eastern Michigan University)
Marisa Exter (Purdue University)
Jonathan Eyolfson (University of Toronto)
Ndudi Ezeamuzie (University of Hong Kong)
Zhiyu Fan (National University of Singapore)
Jixiang Fan (Virginia Polytechnic Institute and State University)
Maryam Fanaeepour (The University of Melbourne)
Maryam Farahmand Asil (Northeastern university)
Mohammed Farghally (Virginia Tech)
Thomas Farmer (University of Pennsylvania)
Alireza Fazelpour (Ventura College)
Pedro Guillermo Feijóo-García (Georgia Institute of Technology)
Alan Fekete (The University of Sydney)
Russell Feldhausen (Kansas State University)
Matthew Ferland (University of Southern California)
Amanda Fernandez (University of Texas at San Antonio)
Nigel Fernandez (University of Massachusetts Amherst)
Deborah Fields (Utah State University)
Gabe Fierro (Colorado School of Mines)
Casey Fiesler (University of Colorado Boulder)
Benjamin Fine (University of Wisconsin - Eau Claire)

Kellyann Fitzpatrick (Georgia Institute of Technology)
Robin Flatland (Siena College)
Scott Fleming (University of Memphis)
Carol Fletcher (UT Austin - TX Advanced Computing Center)
Mark Floryan (University of Virginia)
Daniel Fokum (The University of the West Indies)
Samantha S. Foley (University of Wisconsin-La Crosse)
Morgan Fong (University of Illinois, Urbana-Champaign)
Jack Forden (Marquette)
Amelia Fordyce (Northeastern University)
Randy Fortier (Thompson Rivers University)
Timothy Fossum (Rochester Institute of Technology)
Allan Fowler (University of Auckland)
Max Fowler (University of Illinois Urbana-Champaign)
Susan Eileen Fox (Macalester College)
Pablo Frank Bolton (Smith College)
Diana Franklin (UChicago STEM Education)
Brian Fraser (Simon Fraser University)
Gordon Fraser (University of Passau)
Kathleen Freeman Hennessy (University of Oregon)
Jordan Freitas (Loyola Marymount University)
Michelle Friend (University of Nebraska Omaha)
Rachel Frisbie (Michigan State University)
Jörg Frochte (Bochum University of Applied Sciences)
Hua Leong Fwa (Singapore Management University)
Martin Gagne (Wheaton College)
Sergio Gago (University of California, Irvine)
Leo Galway (Queens University Belfast)
Daniela Ganelin (Stanford University)
Alice Gao (University of Toronto)
Chunming Gao (Central Washington University)
Zhikai Gao (North Carolina State University)
Lisa Garbrecht (The University of Texas at Austin)
Rita Garcia (Victoria University of Wellington)
Dan Garcia (University of California, Berkeley)
Charlie Garrod (Carnegie Mellon University)
David Gary (University of North Carolina at Charlotte)
Adam Gaweda (North Carolina State University)
Alexander Gebhard (Marquette University)
Ed Gehringer (North Carolina State University)
Kari George (University of Illinois Urbana-Champaign)
Yael Gertner (University of Illinois Urbana-Champaign)
Paul Gestwicki (Ball State University)
Abhimanyu Ghosh (University of Wisconsin-Stout)
Ahana Ghosh (Max Planck Institute For Software Systems)
Sudipto Ghosh (Colorado State University)
Nasser Giacaman (The University of Auckland)
David Gibson (Valdosta State University)
David Gillman (New College of Florida)
Sam Gilson (North Carolina State University)
Nandhini Giri (Purdue University)
Olga Glebova (University of Connecticut)
Agoritsa Gogoulou (University of Athens)
Yaroslav Golubev (JetBrains Research)
Elias Gonzalez (University of Maryland)
Javier Gonzalez-Sanchez (California State University)
Daniel Gooch (The Open University)
Martin Goodfellow (University of Strathclyde)
Andrew Goodney (University of Southern California)
Sara Gopalkrishna (George Mason University)
Alex Goslen (North Carolina State University)
Dhiman Goswami (George Mason University)
Michael Gousie (Wheaton College (MA))
Marco Aurélio Graciotto Silva (Federal University of Technology - Paraná)
Isabella Gransbury (North Carolina State University)
Floriana Grasso (University of Liverpool)
Mark Grebe (University of Central Missouri)
Jasmine Greenaway (SIGCSE)
Ross Greer (University of California San Diego)
Chris Gregg (Stanford University)
Luisa Greifenstein (Universität Passau)
Christiane Gresse von Wangenheim (Federal University of Santa Catarina - UFSC)
Andre Greubel (University of Wuerzburg)
William Griswold (University of California San Diego)
Joshua Gross (Blackburn College)
Thomas R. Gross (ETH Zurich)
Charlotte Gruner (Casper College)
Xingjian Gu (University of Michigan)
Giovanna Guerrini (DIBRIS- University of Genova)
Michael Guerzhoy (University of Toronto)
Arup Guha (University of Central Florida)
Anqi Guo (Boston University)
Meize Guo (University of Florida)
Pawan Gupta (University of Virginia)
Yash Gupta (University of Southern California)
Laura Gurney (University of Maine)
Francisco J. Gutierrez (University of Chile)
Juan Gutierrez-Cardenas (Universidad de Lima)
Jesse Ha (Montclair State University)
Steve Hadfield (US Air Force Academy)
Shirin Haji Amin Shirazi (University of California, Riverside)
Georgiana Haldeman (Colgate University)
Lama Hamandi (Northeastern University)
Nadeem Hamid (Berry College)
Susan Hammond (Lipscomb University)
Mikyung Han (The University of Texas at Austin)
Kendrick Hang (Green River College)
Derek Haqq (Virginia Tech)

Caroline Hardin (Western Washington University)
James Harland (RMIT University)
Colton Harper (University of Nebraska-Lincoln)
Neil Harrison (Utah Valley University)
Tim Harrison (Eastern Oregon University)
Mir Hasan (Austin Peay State University)
Muhammad Mustafa Hassan (University of Eastern Finland)
Carl Haynes-Magyar (Carnegie Mellon University)
Raya Hegeman-Davis (University of Wyoming)
Florian Heimerl (University of Wisconsin-Madison)
Cecily Heiner (Oregon Institute of Technology)
Jesse Heines (Univ. of Massachusetts Lowell)
Arto Hellas (Aalto University)
Joelle Henry (CSTA Virginia)
Nicole Herbert (University of Tasmania)
Valeria Herskovic (Pontificia Universidad Católica de Chile)
Hanan Hibshi (Carnegie Mellon University)
Timothy Hickey (Brandeis University)
Alexander Hicks (Virginia Tech)
Matt Higger (Northeastern University)
Timothy Highley (La Salle University)
Lauren Himbeault (University of Manitoba)
Nga Hoang (University of Colorado Boulder)
Janine Hoelscher (University of North Carolina at Chapel Hill)
Chad Hogg (Millersville University)
Amanda Holland-Minkley (Washington & Jefferson College)
Jennifer Holst (John Jay College of Criminal Justice (City University of New York))
Victoria Hong (St. Joseph's University)
Cay Horstmann (San Jose State University)
Diane Horton (University of Toronto)
Melissa Hosten (University of Arizona)
John Hott (University of Virginia)
Jennifer Houchins (WestEd)
Sena Hounsinou (Metro State University)
David Hovemeyer (Johns Hopkins University)
Christopher Hovey (University of Colorado Boulder)
Roberto Hoyle (Oberlin College)
Justin Hsia (University of Washington)
Wen-Jung Hsin (Park University)
Anne Drew Hu (Michigan State University)
Yihao Hu (Duke University)
Fuqun Huang (Institute of Interdisciplinary Scientists)
Ruanqianqian Huang (University of California, San Diego)
Shan Huang (University of Illinois Urbana-Champaign)
Sarah Hug (Colorado Evaluation & Research Consulting)
James Huggins (Kettering University)
Joe Hummel (Northwestern University)
Chris Hundhausen (Oregon State University)
Eping Hung (Southern Oregon University)
Sven Hüsing (Paderborn University)
Deborah Hwang (University of Evansville)
Ahmed Ibrahim (University of Pittsburgh)
Omar Ibrahim (University of Washington)
Elyas Irankhah (University of Massachusetts Lowell)
Kathleen Isenegger (University of Illinois)
Sheikh Rabiul Islam (Rutgers University-Camden)
Md. Tahmidul Islam Molla (Marquette University)
Amy Isvik (North Carolina State University)
Johanna Jacob (The University of Texas at San Antonio)
Miriam Jacobson (University of Texas at Austin)
Mandana Jafarian (Ontario Institute for Studies in Education (OISE), University of Toronto)
Pankaj Jalote (IIIT-Delhi)
Alan Jamieson (Northeastern University)
Lindsay Jamieson (Northeastern University)
Fatemeh Jamshidi (Auburn University)
Vandana Janeja (UMBC)
Ben Jelen (Rose-Hulman Institute of Technology)
Michelle Jensen (University of Wisconsin: Madison)
Minji Jeon (University of Nebraska-Lincoln)
Xiaohua Jia (Northwest Normal University)
Zhen Jiang (West Chester University)
Priscilla Jimenez (Grinnell College)
Adri Jovin John Joseph (Sri Ramakrishna Institute of Technology)
Colin Johnson (University of Nottingham)
Fionnuala Johnson (University of Glasgow)
Michael Johnson (Georgia Institute of Technology)
Alark Joshi (University of San Francisco)
Deepti Joshi (The Citadel)
Ashley Juavinett (UC San Diego)
Maria Jump (Google)
Dilnoza Kadirova (Indiana University Bloomington)
Andrew Kalafut (Grand Valley State University)
Yvonne Kao (WestEd)
Chadi Kari (University of the Pacific)
Deborah Kariuki (UMBC)
Mike Karlin (California State University, Dominguez Hills)
Maria Kasinidou (Open University of Cyprus)
Kate Kastelein (MMSA)
Gloria Katuka (University of Florida)
Anudeep Katukojwala (San Francisco State University)
Erdogan Kaya (George Mason University)
Daniyal Kazempour (Christian-Albrechts-Universität zu Kiel)
Iretta Kearse (Georgia Tech)
Angela Kelling (University of Houston-Clear Lake)

Tyson Kendon (University of Calgary)
Wiliam Kerney (Clovis Community College)
Hieke Keuning (Utrecht University)
Mohammad Taha Khan (Washington and Lee University)
Yekaterina Kharitonova (UCSB)
Natalie Kiesler (DIPF Leibniz Institute for Research and Information in Education)
Daehee Kim (California State University - Stanislaus)
David Young-Jae Kim (Massachusetts Institute of Technology)
Sung Kim (Lewis University)
Jason King (North Carolina State University)
James Kiper (Miami University)
Styliani Kleanthous (CYENS CoE and Open University of Cyprus)
Anthony Kleerekoper (Manchester Metropolitan University)
Devorah Kletenik (Brooklyn College, CUNY)
Joe Kmoch (JK Consulting)
Shao-Heng Ko (Duke University)
Adam Koehler (University of Illinois at Chicago)
Anne Kohlbrenner (Carnegie Mellon University)
Tobias Kohn (TU Wien)
Jack Kolb (University of Minnesota)
Stacey Koornneef (University of Ontario Institute of Technology)
Frank Kreimendahl (Wentworth Institute of Technology)
Shyama Sastha Krishnamoorthy Srinivasan (IIIT-Delhi)
Gerald Kruse (Juniata College)
Bogdan Ksiezopolski (Polish-Japanese Institute of Information Technology)
Ludek Kucera (Charles University)
Amruth Kumar (Ramapo College of New Jersey)
Dhruv Kumar (Indraprastha Institute of Information Technology, Delhi)
Nivedita Kumar (Florida International University)
Viraj Kumar (Indian Institute of Science)
Yulia Kumar (Kean University)
Wanda Kunkle (Penn State Harrisburg)
Zachary Kurmas (Grand Valley State University)
Clifton Kussmaul (Muhlenberg College)
Michael Lachney (Michigan State University)
Dakotah Lambert (Université Jean Monnet)
Ruth Lamprecht (Mount St. Mary's University)
Martina Landman (TU Wien)
Yedidyah Langsam (City University of New York)
Joseph Laracy (Seton Hall University)
Amy Larson (Augsburg University)
Joseph Latessa (Wayne State University)
Edward Latorre (University of Florida)
Brian Law (Illinois Wesleyan University)
Ramon Lawrence (The University of British Columbia)
Ahyoung Lee (Kennesaw State University)
Crystal Lee (MIT)
Janice Lee (The Research Alliance for New York City Schools)
Jeongkyu Lee (Northeastern University)
Norman Lee (Singapore University of Technology and Design)
Wendy Lee (San Jose State University)
Kathleen Lehman (University of California, Los Angeles)
Lukas Lehner (TU Wien)
Rebecca Leis (Full Sail University)
Arun Balajiee Lekshmi Narayanan (University of Pittsburgh)
James Lembke (Milwaukee School of Engineering)
Anna Lenhart (University of Maryland)
Barbara Lerner (Mt. Holyoke College)
Terry Letsche (Wartburg College)
Michael Leverington (Northern Arizona University)
David Levine (Saint Bonaventure University)
Diane Levitt (Cornell Tech)
Colleen Lewis (University of Illinois at Urbana-Champaign)
Cheng Li (University of Science and Technology of China)
Yiwang Li (University of California Riverside)
Cindy Li (Temple University)
Jianhua Li (Deakin University)
Chen Liang (Google LLC)
Jingsai Liang (Westminster University)
Soohyun Nam Liao (University of California San Diego)
Yin-Chan Liao (Georgia State University)
Konstantinos Liaskos (University of Strathclyde)
David Liben-Nowell (Carleton College)
Christine Liebe (Colorado School of Mines)
Mark Liffiton (Illinois Wesleyan University)
Dave Lillethun (Tufts University)
Kevin Lin (University of Washington)
Paul Lin (McDaniel College)
Yuhan Lin (University of Maryland)
Albert Lionelle (Khoury College of Computer Sciences, Northeastern University)
Alex Lishinski (Michigan State University)
David Liu (University of Toronto)
Rongxin Liu (Harvard University)
Yiqian Liu (Texas State University)
Yudong Liu (Western Washington University)
Michael Liut (University of Toronto Mississauga)
David Lockett (Meharry School of Applied Computational Sciences)
Michael Loui (University of Illinois Urbana-Champaign)
Emily Lovell (University of California, Santa Cruz)

Vicente Ferreira De Lucena Jr (Federal University of Amazonas)
Stephanie Lunn (Florida International University)
Shawn Lupoli (Texas A&M University)
Collin Lynch (North Carolina State University)
Qi Lyu (Michigan State University)
Mirella M. Moro (Univ. Federal de Minas Gerais - UFMG)
Faten M'Hiri (Mcgill university)
Cam Macdonell (MacEwan University)
Sean Mackay (University at Buffalo)
Vaishali Mahipal (University of Massachusetts Lowell)
Maryam Majedi (University of California, Santa Barbara)
Janice Mak (Arizona State University)
Fathimath Zuha Maksood (Carnegie Mellon University)
Ali Malik (Stanford University)
Eric D. Manley (Drake University)
Rifat Sabbir Mansur (Virginia Tech)
George Maratos (University of Illinois at Chicago)
Venkat Margapuri (Villanova University)
Lauren Margulieux (Georgia State University)
Thomas Marlowe (Seton Hall University)
Fred Martin (University of Texas at San Antonio)
Patrick Martin (University of Richmond)
Samiha Marwan (University of Virginia)
Raina Mason (Southern Cross University)
Stephanos Matsumoto (Olin College of Engineering)
Suzanne Matthews (United States Military Academy)
Stephanos Mavromoustakos (Indiana Tech)
Bruce Maxwell (Colby College)
Eric Mayhew (Dawson College)
Samuel Mazzone (Marquette University)
Chao Mbogho (Mozilla Foundation)
Paul Mcburney (West Virginia University)
Chris McCaig (University of Glasgow)
Lester McCann (University of Arizona)
Declan McClintock (Michigan State University)
Robert McCloskey (University of Scranton)
Frank Mccown (Harding University)
Sean McCulloch (Ohio Wesleyan University)
Melinda Mcdaniel (Georgia Institute of Technology)
Chris McDonald (The Univ. of Western Australia)
Monica McGill (Institute for Advancing Computing Education)
James McGuffee (Christian Brothers University)
Sterling McLeod (North Carolina State University)
Fiona McNeill (University of Edinburgh)
Brian McSkimming (University of Oklahoma)
António Mendes (University of Coimbra)
Amanda Menier (SageFox Consulting Group)
Larry Merkle (Air Force Institute of Technology)
Marcus Messer (King's College London)
Nazmus Sakib Miazi (Northeastern University)
Tilman Michaeli (TU Munich)
Daphne Miedema (Eindhoven University of Technology)
Radu Mihail (Valdosta State University)
Heather Miles (Gateway Technical College)
Michael Miljanovic (Ontario Tech University)
Craig S. Miller (DePaul University)
Mark Miller (Learningtech.org)
Afsaneh Minaie (Utah Valley University)
Marius Minea (Politehnica University of Timisoara)
Mia Minnes (University of California San Diego)
Joydeep Mitra (Stony Brook University)
Simanta Mitra (ISU)
Tanja Mitrovic (Intelligent Computer Tutoring Group, University of Canterbury, Christchurch)
Mohamed Wiem Mkaouer (Rochester Institute of Technology)
Idongesit Mkpong-Ruffin (Florida A&M University)
Kenrick Mock (University of Alaska Anchorage)
Micah G. Modell (Mount Saint Mary College)
Heng Ngee Mok (Singapore Management University)
William Mongan (Drexel University)
Alvaro Monge (California State University Long Beach)
Igor Dos Santos Montagner (Insper)
Hannah Moon (Georgia Institute of Technology)
Firas Moosvi (University of British Columbia Okanagan)
Marcia Moraes (Colorado State University (CSU))
Luis Morales-Navarro (University of Pennsylvania)
Marco T. Morazan (Seton Hall University)
Hannah Morgan (University of Chicago)
Patricia Morreale (Kean University)
Karl A. Morris (Temple University)
Matthew Morrison (University of Notre Dame)
Bradford Mott (North Carolina State University)
Bouzazi Mouna (Institut superieur de gestion de tunis)
Murshida Rahman Mouree (University of Memphis)
Chad Mourning (Ohio University)
Sai Mukkavilli (Georgia Southwestern State University)
Kasia Muldner (Carleton University, Canada)
Tony Mullen (Northeastern University)
Christian Murphy (Swarthmore College)
Laurie Murphy (Pacific Lutheran University)
Ilya Musabirov (University of Toronto)
Line Have Musaeus (Aarhus University)
Dave Musicant (Carleton College)
Raelin Musuraca (Carnegie Mellon University)
Felix Muzny (Northeastern University)
Timothy Myatt (Trevecca Nazarene University)
Nomair Naeem (University of Waterloo)
Varun Nagaraj Rao (Princeton University)
Nadia Najjar (University of North Carolina at Charlotte)

Nannette Napier (Georgia State University)
Sashank Narain (University of Massachusetts Lowell)
Krish Narayanan (Eastern Michigan University)
Nabeel Nasir (University of Virginia)
Sagnik Nath (University of California, Santa Cruz)
Tamara Nelson-Fromm (University of Michigan)
Chantra Nhien (University of California Los Angeles)
Christopher Nitta (University of California, Davis)
Manee Ngozi Nnamani (Massachusetts Institute of Technology)
Ridwan Rashid Noel (Texas Lutheran University)
Emma Norling (The University of Sheffield)
Victor Norman (Calvin College)
Ed Novak (Franklin and Marshall College)
Dr. Nasheen Nur (Florida Institute of Technology)
Sara Nurollahian (University of Utah)
Emily Nutwell (The Ohio State University)
Brian O'Neill (Quinnipiac University)
Suhaib Obeidat (Shenandoah University)
Sheila Oh (Seattle University)
Peter Ohmann (College of St. Benedict / St. John's University)
Vidushi Ojha (University of Illinois at Urbana-Champaign)
Omofolakunmi Olagbemi (Hope College)
Viktoriya Olari (Freie Universität Berlin)
Priti Oli (University of Memphis)
Huey Fang Ong (Monash University Malaysia)
Ethan Ordentlich (University of Illinois Chicago)
Zachary J. Oster (University of Wisconsin-Whitewater)
Aadarsh Padiyath (University of Michigan, Ann Arbor)
Arthi Padmanabhan (Harvey Mudd College)
Merril Paez (Western Washington University)
Christopher Painter-Wakefield (Colorado School of Mines)
Katherine Panciera (Milwaukee School of Engineering)
Ayush Pandey (University of California, Merced)
Ashley Pang (University of California, Riverside)
Maciej Pankiewicz (Warsaw University of Life Sciences)
Asma Paracha (Seneca Polytechnic)
Jennifer Parham-Mocello (Oregon State University)
Sunjae Park (Wentworth Institute of Technology)
Youngki Park (Chuncheon National University of Education)
Jack Parkinson (University of Glasgow)
P D Parthasarathy (BITS Pilani, KK Birla Goa Campus)
Muhammad Qasim Pasta (PAF -KIET)
Nick Patrikeos (University of New South Wales)
Brian Patterson (Oglethorpe University)
Daniel Patterson (Northeastern University)
Tina Pavlovich (Dartmouth College)
Emrah Pektas (University of Massachusetts Amherst)
Radek Pelánek (Masaryk University Brno)
Joslenne Pena (Macalester College)
Bin Peng (Park University)
Bernardo Pereira Nunes (Australian National University)
Melissa Perez (University of Michigan)
Manuel Perez-Quinones (University of North Carolina, Charlotte)
Corrina Perrone Smith (AgentSheets, Inc.)
Kathryn Perry (Burnt Hills - Ballston Lake High School)
Andrew Petersen (University of Toronto)
Luigia Petre (Åbo Akademi University)
Robert Pettit (The Aerospace Corporation)
Jeffrey Pfaffmann (Lafayette College)
Stephen Piccolo (Brigham Young University)
Charlotte Pierce (Monash University)
Apivadee Piyatumrong (National Electronics and Computer Technology Center (NECTEC))
Brian Plancher (Barnard College, Columbia University)
Shari Plantz-Masters (Regis University)
Dr. Md Masbaul Alam Polash (The University of Sydney)
Jack Polifka (Boise State University)
Joe Politz (University of California San Diego)
Tommy Pollock (Tidewater Community College)
Jeffrey Popyack (Drexel University)
Joël Porquet-Lupine (University of California, Davis)
Lori Postner (Nassau Community College)
Seth Poulsen (Utah State University)
Laurel Powell (University of North Carolina at Charlotte)
Prajish Prasad (FLAME University)
Adam Purtee (University of Rochester)
Yizhou Qian (Jiangnan University)
Meiying Qin (York University)
James Quinlan (University of Southern Maine)
Mohammed Saferi Rahman (AI)
Md Nishat Raihan (George Mason University)
Brian Railing (Carnegie Mellon University)
Maryam Raiyat Aliabadi (University of British Clumbia)
Siva Raj Rajagopalan (New York University)
Sujeeth Goud Ramagoni (Marquette University)
Arun Raman (BITS Pilani, Goa)
Harini Ramaprasad (UNC Charlotte)
Timothy Randolph (Columbia University)
Yolanda Rankin (Emory University)
Matthew Rattigan (University of Massachusetts Amherst)
Veena Ravishankar (University of Mary Washington)
Pranathi Rayavaram (University of Massachusetts Lowell)
Ali Raza (University of Colorado Boulder)
Syeda Saleha Raza (Habib University)
Saquib Razak (University of Michigan)
Mimi Recker (utah state university)
Shanon Reckinger (University of Illinois at Chicago)
Michael Redmond (La Salle University)

Benjamin Reed (San José State University)
Dale Reed (University of Illinois Chicago)
David Reed (Creighton University)
Donna Reese (Mississippi State University)
Brent Reeves (Abilene Christian University)
Heidi Reichert (North Carolina State University)
Karen Reid (University of Toronto)
Charles Reiss (University of Virginia)
Jed Rembold (Willamette University)
He Ren (University of Washington)
Margherita Renieri (IMT School for Advanced Studies Lucca)
Jake Renzella (University of New South Wales, Sydney)
Cheryl Resch (University of Florida)
Mohi Reza (University of Toronto)
Junghwan Rhee (University of Central Oklahoma)
Leila Ribeiro (Universidade Federal do Rio Grande do Sul)
Mike Richards (The Open University)
Vance Ricks (Northeastern University)
Charles Riedesel (University of Nebraska - Lincoln)
Philip Ritchey (Texas A&M University)
Anna Ritz (Reed College)
Kelly Rivers (Carnegie Mellon University)
Nouhad Rizk (University of Houston)
Michael Robbeloth (Mount Vernon Nazarene University (MVNU))
Henry Robbins (Cornell University)
Christian Roberson (Florida Southern College)
Cindy Robertson (Georgia Gwinnett College)
Andey Robins (University of Wyoming)
Rachel Roca (Michigan State University)
Hemilis Rocha (UFPE)
Gustavo Rodriguez-Rivera (Purdue University)
Phil Romig (Colorado School of Miens)
Nicole Ronald (Swinburne University of Technology)
Darcy Ronan (Sacred Heart University)
Kevin Ros (University of Illinois at Urbana-Champaign)
Nicholas Rosasco (Valparaiso University)
Jennifer Rosato (University of Minnesota)
Nimisha Roy (Georgia Institute of Technology)
Tania Roy (New College of Florida)
Ahana Roy Choudhury (Valdosta State University)
Sean Russell (University College Dublin)
Dana Saito-Stehberger (University of California, Irvine)
Majd Sakr (Carnegie Mellon University)
Amin Sakzad (Monash University)
Waqar Saleem (Habib University)
Adrian Salguero (UC San Diego)
Abdul Samad (BUITEMS)
Maria Sanchez (University of Maryland Baltimore County)
Mario Sanchez (Universidad de los Andes)
Smaranda Sandu (Wellesley College)
Julio Santisteban (Universidad Católica San Pablo)
Eddie Antonio Santos (University College Dublin)
Nurani Saoda (University of Virginia)
Ankit Satpute (FIZ Karlsruhe - Leibniz-Institut für Informationsinfrastruktur)
Allison Sauppé (University of Wisconsin-La Crosse)
Jaromir Savelka (Carnegie Mellon University)
Benjamin Say (Metropolitan Community College)
Fernando Schapachnik (Universidad de Buenos Aires, Buenos Aires, Argentina)
Bruce Scharlau (University of Aberdeen)
Colin Schatz (Independent)
Jule Schatz (University of Illinois Urbana Champaign)
Lara Schenck (Georgia Institute of Technology)
Herman Schinca (Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires)
Jessica Young Schmidt (North Carolina State University)
Karl Schmitt (Trinity Christian College)
Jonatan Schroeder (York University)
Rebecca Schroeder (University of Texas at San Antonio)
Cory Scott (Colorado College)
Michael Scott (Falmouth University)
Vibhor Sehgal (University of California, Berkeley)
Manuel Selva (INRIA)
Cigdem Sengul (Brunel University)
Sue Sentance (University of Cambridge)
Christian Servin (El Paso Community College)
Carolyn Seton (Southern Cross University)
Amber Settle (DePaul University)
Zulal Sevkli (Miami University)
Mohammed Seyam (Virginia Tech)
Jeffrey Shafer (University of the Pacific)
Anshul Shah (University of California, San Diego)
Bhumika Shah (Department of Computer Science)
Selina Marianna Shah (Amrita Vishwa Vidyapeetham)
Sheikh Ahmad Shah (Boston College)
Rishabh Sharma (Hewlett Packard Enterprise)
Sadia Sharmin (University of Toronto)
Erin Shaw (Pasadena City College)
Umar Shehzad (Utah State University)
Josh Sheldon (MIT App Inventor)
Mark Sherriff (University of Virginia)
Heather Sherwood (Education Development Center)
Yang Shi (North Carolina State University)
Jon Shidal (Washington University in Saint Louis)
Katie Shilton (University of Maryland)
Michael Shindler (University of California, Irvine)
Yasuto Shirai (Shizuoka University)
Lindsey Shorser (University of Toronto)
Ankit Shrestha (Utah State University)

Naaz Sibia (University of Toronto)
Bill Siever (Washington University)
Leonardo Silva (IFPE)
Mariana Silva (University of Illinois at Urbana-Champaign)
Gabriel Silva de Oliveira (North Carolina State University)
Mario Simaremare (Institut Teknologi Del)
Bogdan Simion (University of Toronto)
Simon (Unaffiliated)
Anjali Singh (University of Michigan)
Lenandlar Singh (University of Guyana)
Sarita Singh (Northeastern University)
James Skripchuk (North Carolina State University)
David Smith (University of Illinois Urbana-Champaign)
Estelle Smith (Colorado School of Mines)
Jacqueline Smith (University of Toronto)
Julie Smith (CSEdResearch.org)
Katie Smith (Temple University)
Mary Smith (HPU)
Monika Sobolewska (University of Washington - Tacoma)
Jaemarie Solyst (Carnegie Mellon University)
Yifan Song (University of Illinois Urbana-Champaign)
Raja Sooriamurthi (Carnegie Mellon University)
Madalene Spezialetti (Trinity)
Joseph Spring (University of Hertfordshire)
Scott Spurlock (Elon University)
Meera Sridhar (University of North Carolina Charlotte)
Andreas Stefik (University of Nevada, Las Vegas)
Phil Steinhorst (University of Münster)
Alexander Steinmaurer (Graz University of Technology)
Tom Stephens (Brigham Young University)
Kristin Stephens-Martinez (Duke University)
Ben Stephenson (University of Calgary)
Sandeep Sthapit (North Carolina State University)
Kaitlyn Stormes (University of California, Los Angeles)
John Stratton (Whitman College)
Sven Strickroth (LMU Munich)
Kristina Striegnitz (Union College)
Amber Stubbs (Simmons College)
Sneha Sudhakaran (Florida Institute of Technology)
Kuei Sun (University of Toronto)
Timothy Sun (San Francisco State University)
Sara Sutton (Assistant Profesor)
Oleg Sychev (Volgograd State Technical University)
Benyamin T. Tabarsi (North Carolina State University)
Sri Yash Tadimalla (University of North Carolina Charlotte)
Sareh Taebi (Florida Atlantic University)
Ruby Tahboub (University of Illinois Urbana-Champaign)
Robert Tairas (Vanderbilt University)
Edward Talmage (Bucknell University)
Jialiang Tan (Lehigh University)
Sheng Tan (Trinity University)
Patricia Tanner (Indiana Tech)
Md. Hasan Tareque (University of Lethbridge)
Kamilla Tenório (Freie Universität Berlin)
Kia Teymourian (The University of Texas at Austin)
Kyle Thayer (University of Washington)
George Thomas (University of Wisconsin Oshkosh)
Madison Thomas (North Carolina State University)
Cynthia Thomas (Northern Kentucky University)
Michael Thompson (Central College)
Sverrir Thorgeirsson (ETH Zürich)
Carsten Thue-Bludworth (Reviewer for SIGSCE24)
Xiaoyi Tian (University of Pittsburgh)
Sarina Till (University of Cape Town)
Joseph Tise (CSEdResearch.org)
Sirazum Munira Tisha (Rollins College)
Hapnes Toba (Maranatha Christian University)
Ryan Torbey (The University of Texas at Austin)
Lisa Torrey (St. Lawrence University)
Giulia Toti (University of British Columbia)
Matthew Toups (Tulane University)
Mohammad Toutiaee (Northeastern University)
Jelena Trajkovic (California State University Long Beach)
Christopher Tralie (Duke University)
Keith Tran (NC State University)
Christian Trefftz (Grand Valley State University)
Yun-Cheng Tsai (National Taiwan Normal University)
Yu-Cheng Tu (The University of Auckland)
Jeffrey Turkstra (Purdue University)
Wesley Turner (Rensselaer Polytechnic Institute)
Guenter Tusch (Grand Valley State University)
Ella Tuson (Brandeis University)
Malibongwe Twani (Nelson Mandela University)
Viktor Uglev (Siberian Federal University)
R.N. Uma (North Carolina Central University)
Leo Ureel II (Michigan Technological University)
Jan Vahrenhold (Westfälische Wilhelms-Universität Münster)
Scott Valcourt (Northeastern University)
Andrew Valentine (University of Melbourne)
Robert Van Camp (Marietta College)
Andrew van der Poel (Northeastern University)
Jessica Vandenberg (North Carolina State University)
James Vanderhyde (Saint Xavier University)
Tullio Vardanega (University of Padua)
Pooja Vashisth (York University)
Alexandra Vassar (UNSW, Sydney)
Nanette Veilleux (Simmons College)
Yesenia Velasco (Duke University)
Krishna Vellamchety (University of Massachusetts Lowell)
Sukrit Venkatagiri (Swarthmore College)

Rajagopal Venkatesaramani (Washington University in St. Louis)
Jonathan Ventura (California Polytechnic State University San Luis Obispo)
Michael Verdicchio (The Citadel, The Military College of South Carolina)
Lindsey Vetvick (Discovery Partners Institute)
Isabella Vitale (Adelphi University)
Amy Voida (University of Colorado Boulder)
Jan Vykopal (Masaryk University)
Amber Wagner (University of Alabama at Birmingham)
Jane Waite (Raspberry Pi Foundation)
David Wakeling (University of Exeter)
Bettina Waldvogel (University of Teacher Education Schwyz)
Justice Walker (University of Pennsylvania)
Charles Wallace (Michigan Technological University)
Hsiao-An Wang (Northeastern University)
Matt Wang (University of Washington)
Patrick Wang (Haute École Pédagogique du canton de Vaud)
Yang Wang (La Salle University)
Yi Wang (Beijing University of Posts and Telecommunications)
Zhuoying Wang (The University of Texas at Austin)
Yuanxin Wang (Carnegie Mellon University)
Kevin Wayne (Princeton University)
Jason Weber (University of California, Irvine)
Robbie Weber (University of Washington)
Michael Weeks (Georgia State University)
Karl-Heinz Weidmann (Universtiy of Applied Sciences Vorarlberg)
Pierre Weill-Tessier (King's College London)
Richard Weiss (The Evergreen State College)
Stephanie Werner (University of Illinois, IWERC)
Jacqueline Whalley (Auckland University of Technology)
Guy E White (Harvard CS50)
Gregory White (UTSA/CIAS)
Howard Whitston (University of South Alabama)
Robert Whyte (Raspberry Pi Foundation)
Jina Wilde (University of Texas at San Antonio)
Chad Williams (Central Connecticut State University)
Jordan Williamson (CSEd research)
Jaxton Winder (Utah State University)
Gunnar Eyal Wolf Iszaevich (Instituto de Investigaciones Económicas UNAM)
Steven Wolfman (The University of British Columbia)
Jennifer Wong-Ma (UC Irvine)
Ira Woodring (Grand Valley State University)
Stephanie Wortel-London (CSforALL)
Brett Wortzman (University of Washington)
John Wright (Juniata College)
Melissa Wright (Barnard College)
Leanne Wu (University of Calgary)
Wensheng Wu (University of Southern California)
Zihan Wu (University of Michigan)
Burkhard Wuensche (The University of Auckland)
Karl Wurst (Worcester State University)
Franceska Xhakaj (Carnegie Mellon University)
Maia Xiao (University of Washington)
Zhiguang Xu (Valdosta State University)
Haoru Xue (Carnegie Mellon University)
Gautam Yadav (Carnegie Mellon University)
Ryoko Yamaguchi (Plus Alpha Research & Consulting LLC)
Shenao Yan (University of Connecticut)
Wei Yan (University of Florida)
Hui Yang (SRI International)
Ruhan Yang (University of Colorado Boulder)
Arthur Yanushka (Christian Brothers University)
Jessica Yauney (Brigham Young University)
Tom Yeh (UC Irvine)
Chansu Yu (Cleveland State University)
Jones Yu (Wentworth Institute of Technology)
Tianwei Yue (Independent)
David Zabner (Cornell College)
Daniel Zappala (Brigham Young University)
Hedayat Zarkoob (University of British Columbia)
Armita Zarnegar (Swinburne University of Technoloyg)
Angela Zavaleta Bernuy (University of Toronto)
Michael Zawatski (University of Texas San Antonio)
Julie Zelenski (Stanford University)
Huiyu Zhang (Temasek Polytechnic)
Larry Yueli Zhang (York University)
Liang Zhang (Beihang University)
Lisa Zhang (University of Toronto)
Yifan Zhang (University of Delaware)
Yiji Zhang (Rose-Hulman Institute of Technology)
Jia Zhu (Florida International University)
Tina Zhu (Greenwich Academy)
Tingting Zhu (University of Toronto Mississauga)
Yong Zhuang (Grand Valley State University)
Craig Zilles (University of Illinois at Urbana-Champaign)
Hadar Ziv (University of California, Irvine)

SIGCSE TS 2024 Sponsors & Supporters

Sponsors





Platinum Supporters







Gold Supporters







Silver Supporters





AI in Computing Education from Research to Practice

Bita Akram
North Carolina State University
Raleigh, North Carolina, USA
bakram@ncsu.edu

Juho Leinonen
Aalto University
Espoo, Finland
juho.2.leinonen@aalto.fi

Narges Norouzi
University of California Berkeley
Berkeley, California, USA
norouzi@berkeley.edu

James Prather (Moderator)
Abilene Christian University
Abilene, Texas, USA
jrp09a@acu.edu

Lisa Zhang
University of Toronto
Toronto, Ontario, Canada
lczhang@cs.toronto.edu

ABSTRACT

The panel comprises a diverse set of Computing educators working on AI in education. The panelists will address four areas of AI in Computing education: 1) AI for introductory CS classrooms, 2) Investigating opportunities presented by LLMs, 3) LLM-based tool development, and 4) Ethics and inclusion in AI curriculum. The panel will share experiences and discuss opportunities and challenges in AI education with the community.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

Artificial intelligence, large language models, computing education

ACM Reference Format:
Bita Akram, Juho Leinonen, Narges Norouzi, James Prather (Moderator), and Lisa Zhang. 2024. AI in Computing Education from Research to Practice. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3631657

1 SUMMARY

The emergence of advanced Artificial Intelligence (AI) technologies, including generative AI approaches, brought ample opportunities and challenges in Computing education. While educators have the freedom to integrate AI tools and curricular content within their classrooms, classes will inevitably face unprecedented changes.

In this panel, panelists will discuss their experience and vision for including AI-focused learning technologies and curriculum and their perceived challenges and opportunities this integration presents. They discuss effective research methodologies to ensure proper design, implementation, and evaluation of AI-based technology and curriculum.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3631657

2 PANEL STRUCTURE

The panel comprises early-career Computing educators with diverse backgrounds from three different countries. Collectively, the panel will share their experiences and insights on AI in Computing education from research to practice. The panelists would like to use this opportunity to share opportunities and challenges of adopting AI, especially Large Language Models (LLMs), in Computing education. We also plan to use the platform to hear the audience's opinions and concerns and brainstorm avenues to address them as a community. Since the topic is broad and requires a culmination of expertise, the four panelists will discuss their perspectives for 10 minutes each, leaving 35 minutes for audience Q&A.

3 POSITION STATEMENTS

3.1 Bita Akram: AI for Introductory CS Classrooms

Bita Akram is an Assistant Research Professor at North Carolina State University with extensive experience in research and practice in introductory programming classrooms. She leads multiple NSF-funded projects for designing an AI-enabled classroom assistant for enhancing instruction in introductory programming courses [6, 7] and fostering persistence among underrepresented students in Computing [1, 5]. With the recent advances in AI technology and the drastic increase in CS enrollments, it is imperative to leverage the advantages of emerging AI technologies to foster novice students' learning by offering scalable and adaptive pedagogical support. The advances in AI not only entail the need to enhance introductory Computing education through adaptive technology but also pronounce the importance of integrating AI as an integral part of the introductory CS curriculum. The early introduction to opportunities and challenges provided by AI can positively affect students' career aspirations about AI and bring about the imperative awareness of ethical and responsible AI.

3.2 Juho Leinonen: Investigating the Opportunities Presented by LLMs in Computing Education: Research and Integration

Juho Leinonen is an Academy Research Fellow at Aalto University and has researched the potential opportunities that LLMs provide for Computing education, including giving a keynote and leading a working group on the topic at ITiCSE 2023. His research has found

many applications of LLMs for Computing education, ranging from enhancing programming error messages [9] to automatic creation of personalized programming exercises [15] and natural language explanations of source code [8]. Integrating LLMs into Computing education practice could revolutionize how Computing is taught. At this critical juncture, research on LLMs and their integration into teaching is not just a trend but a necessity. Prohibiting the use of AI in Computing education is unfeasible, as models are freely available, and students are already using them. Thus, we should not only integrate LLMs into teaching but actively teach students how to use them productively while highlighting their limitations, such as the potential to create biased content and the tendency to hallucinate. Embracing this technological advancement is crucial to preparing students for the generative AI era.

3.3 Narges Norouzi: LLM-based Tool Development in Computing Education

Recently, LLMs have been utilized to provide innovative solutions that enrich students' learning experiences. These tools span a variety of applications, such as personalized programming assistance, instant code analysis, comprehensive explanations, question generation, and chatbots. Narges Norouzi is an Assistant Teaching Professor at the University of California, Berkeley. She has been involved in building instructional technology that utilizes AI for intelligence tutoring and adaptive grading systems [11, 14]. She has recently focused on developing and deploying LLM-based instructional technologies in two main areas: 1) providing real-time tutoring support to students in CS1 [16] and 2) a browser extension tool for facilitating responses to student questions on course discussion forums [10, 12]. On this panel, she will share her experience building LLM-based tools in Computing education and associated challenges. She will also share with the community the potential avenues for further utilization of artificial intelligence, particularly LLMs, to improve Computing education.

3.4 James Prather (Moderator)

James Prather has been a researcher at the intersection of Human-Computer Interaction and Computing education since 2017. His previous work focused on programming error messages and novice programmer metacognition. Since early 2022, he has worked on the impact of large language models in Computing education [2, 3, 9]. He has given keynote talks and was involved in related ITiCSE working groups spanning all issues that will be discussed on the panel, and as such, is a suitable moderator to lead the conversation.

3.5 Lisa Zhang: Ethics and Inclusion in AI Curriculum for Computing

Lisa Zhang is an Assistant Professor Teaching Stream at the University of Toronto Mississauga. She has taught various machine learning courses since 2018 and was named one of the EAAI New and Future AI Educators in 2020. She developed and taught the inaugural offering of Artificial Intelligence Fundamental, which teaches deep learning concepts with little math. She has developed many experiential learning labs for various courses, many of which are accepted into EAAI's Model AI assignments [4, 13]. She is interested in ways to structure machine learning courses to support a range of learners [17].

As AI becomes more relevant in our society, it is now more important than ever to consider how AI education approaches help or hinder broadening participation in the field. Identifying and mitigating barriers and challenges that students face in AI is an approach to broadening participation. For example, one barrier typically vocalized by students is the math requirements in theoretical ML courses, so supporting students through reviewing mathematics and statistics prerequisites could be an approach to reducing the impact of this barrier.

REFERENCES

[1] Bita Akram, Susan Fisk, Spencer Yoder, Cynthia Hunt, Thomas Price, Lina Battestilli, and Tiffany Barnes. 2022. Increasing Students' Persistence in Computer Science through a Lightweight Scalable Intervention. In *ITiCSE'22*.

[2] Brett A Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming Is Hard–Or at Least It Used to Be: Educational Opportunities And Challenges of AI Code Generation. (2023).

[3] James Finnie-Ansley, Paul Denny, Brett A Becker, Andrew Luxton-Reilly, and James Prather. 2022. The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference*.

[4] Michael Guerzhoy, Lisa Zhang, and Georgy Noarov. 2019. AI education matters: building a fake news detector. *AI Matters* 5, 3 (2019).

[5] Rachel Harred, Tiffany Barnes, Susan R Fisk, Bita Akram, Thomas W Price, and Spencer Yoder. 2023. Do Intentions to Persist Predict Short-Term Computing Course Enrollments: A Scale Development, Validation, and Reliability Analysis. In *SIGCSE'23*.

[6] Muntasir Hoq, Peter Brusilovsky, and Bita Akram. 2023. Analysis of an Explainable Student Performance Prediction Model in an Introductory Programming Course. In *EDM'23*.

[7] Muntasir Hoq, Sushanth Reddy Chilla, Melika Ahmadi Ranjbar, Peter Brusilovsky, and Bita Akram. 2023. SANN: Programming Code Representation Using Attention Neural Network with Optimized Subtree Extraction. In *CIKM'23*.

[8] Juho Leinonen, Paul Denny, Stephen MacNeil, Sami Sarsa, Seth Bernstein, Joanne Kim, Andrew Tran, and Arto Hellas. 2023. Comparing Code Explanations Created by Students and Large Language Models. In *ITiCSE'23*.

[9] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A Becker. 2023. Using Large Language Models to Enhance Programming Error Messages. In *SIGCSE'23*.

[10] M. Miroyan, C. Mitra, R. Jain, V. Kumud, G. Ranade, and Narges Norouzi. 2024. RetLLM-E: Retrieval-Prompt Strategy for Question-Answering on Student Discussion Forums. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[11] M. Miroyan, S. Weng, R. Shah, L. Yan, and Narges Norouzi. 2024. EIT: Earnest Insight Toolkit for Evaluating Students' Earnestness in Interactive Lecture Participation Exercises. In *SIGCSE'24*.

[12] C. Mitra, M. Miroyan, R. Jain, V. Kumud, G. Ranade, and Narges Norouzi. 2024. Elevating Learning Experiences: Leveraging Large Language Models as Student-Facing Assistants in Discussion Forums. In *SIGCSE'24 V. 2*.

[13] Todd W Neller, Stephen Keeley, Michael Guerzhoy, Wolfgang Hoenig, Jiaoyang Li, Sven Koenig, Ameet Soni, Krista Thomason, Lisa Zhang, Bibin Sebastian, et al. 2020. Model AI Assignments 2020. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34.

[14] Narges Norouzi and Amir Mazaheri. 2023. Context-Aware Analysis of Group Submissions for Group Anomaly Detection and Performance Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37.

[15] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic generation of programming exercises and code explanations using large language models. In *ICER'22*.

[16] J.D. Zamfirescu-Pereira, L. Qi, B. Hartmann, J. DeNero, and Narges Norouzi. 2024. Conversational Programming with LLM-Powered Interactive Support in an Introductory Computer Science Course. In *NeurIPS'23 Workshop: Generative AI for Education (GAIED)*.

[17] Lisa Zhang and Sonya Allin. 2023. Just-In-Time Prerequisite Review for a Machine Learning Course. In *Proceedings of the 25th Western Canadian Conference on Computing Education*.

Managing Authority When Teaching Computing Ethics

Emmanuelle Burton
University of Illinois Chicago
Chicago, Illinois, USA
enburton@uic.edu

Kristel Clayville
University of Illinois Chicago
Chicago, Illinois, USA
kclayvil@uic.edu

Stacy A. Doore
Colby College
Waterville, Maine, USA
sadoore@colby.edu

Michael S. Kirkpatrick
James Madison University
Harrisonburg, Virginia, USA
kirkpams@jmu.edu

Michael Goldweber
Denison University
Granville, Ohio, USA
goldweberm@denison.edu

ABSTRACT

This panel will help CS-trained educators who are teaching students about how to approach their work with an ethical mindset by discussing one of the key challenges in CS ethics education: how to manage authority in a classroom that focuses on values and ethics. Many CS instructors are wary of teaching either a dedicated ethics course or embedding responsible computing concepts and practices into their core CS courses because they feel that they lack disciplinary authority, or are even unsure of what precise teaching outcomes to aim for. These concerns are real: ethical reasoning is not the same type of skill set as is required in design or engineering work; it is a set of practices rooted in a person's beliefs, judgments, decisions, and values and it is necessarily personal for anyone who takes it seriously. Effective ethics course pedagogy and content are primarily about developing the students' capacity for reflection and inquiry, not transmitting knowledge. In an ethical reasoning focused learning environment, the majority of the work is persuading students that they can and should keep caring and thinking about the impact of their work beyond the classroom: in other words, the students must feel responsible for the reasoning and choices they make that will impact the lives of others. Through a discussion with multiple CS ethics educators, this panel will offer practical approaches to managing authority in this type of classroom as well as suggestions for engaging and empowering students to be curious and reflective about their own values and ethical positions.

CCS CONCEPTS

• **Social and professional topics → Computational science and engineering education**; **Codes of ethics**.

KEYWORDS

ethics, responsible computing pedagogy, authority, professional identity

ACM Reference Format:
Emmanuelle Burton, Kristel Clayville, Stacy A. Doore, Michael S. Kirkpatrick, and Michael Goldweber. 2024. Managing Authority When Teaching Computing Ethics. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3631659

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE '24, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3631659

1 SUMMARY

Most computer science departments have recognized that ethics needs to be a part of the CS training and professional preparation, even though the structure of this type of instruction and inquiry may differ from core CS courses. Nonetheless, many CS educators don't feel confident teaching ethics themselves, precisely because of these differences. This panel aims to help close that gap for CS-trained educators who are teaching CS ethics (or considering it) through a focused conversation among experienced computing ethics educators on the subject of how they manage authority when they are discussing ethics and responsible computing principles with students: their own authority as instructors; the students' authority as people with their own values, experiences and capacity for judgment; and the authority of different kinds of ground truth that people appeal to when making ethical arguments. By reflecting on these different sites of authority – and by including students in these reflections – instructors will be better equipped to help students grasp the full breadth of what it means to be ethically responsible citizens of their profession and of the world, and to persuade them to commit to it.

This type of persuasion is uniquely important when teaching ethics, because it's very unlikely that students' future jobs will hold them meaningfully accountable for making reflective, ethically responsible choices. In fact, large portions of the industry still operates from a "move fast and break things" mindset without concern for the consequences. New technologists face an environment that more highly values profit shares, media recognition, and time to market than the ability to reason through the harms or benefits of their professional actions. How can we equip our students with the capacity to take responsibility for the impact of their work, even when their work environment likely will not? And, to that end: how do we get comfortable teaching from a position that says, "I don't have the answers here, you need to figure out what your answers are and why before you get out there"?

2 PANEL STRUCTURE

After brief introductions from each panelist, the time will be evenly split between a chair-guided roundtable discussion among panelists which will pause for questions from the audience.

2.1 Position Statements

2.2 Emanuelle Burton (panel chair)

Ethics education makes distinctive demands of us as educators. A key part of our pedagogical task is to elicit (or inculcate) a sense of responsibility in our students that they will carry forward, which requires that we create a classroom environment in which they feel themselves meaningfully accountable for making good decisions, but which also challenges them to reflect deeply on the judgments and perceptions that inform their own decision-making. In order to create this environment, an instructor needs to be thoughtful and intentional about what kinds of authority they claim (and what kinds they disclaim) while teaching ethics.

Emanuelle Burton is a co-author of Computing and Technology Ethics: Engaging Through Science Fiction (MIT Press, 2023.) She holds a PhD in Religion and Literature from the University of Chicago, and is a lecturer in the computer science department at the University of Illinois Chicago. She is an active member of ACM's Committee on Professional Ethics.

2.3 Kristel Clayville

There are multiple ways to conceptualize the authority of the teacher in a classroom that is intentionally including ethical material. The teacher can position themselves as a knower of information; a role model; a dispassionate objective thinker; a person grappling with the same questions as the students; or any mix of these. Each of these positions constructs authority differently and implies its own ethical model. Within many technical CS courses, students bring in specific models of authority that they anticipate a teacher to embody, namely, the knower of information and the dispassionate objective thinker, and it can be hard for students and teachers alike to shift models when conversation shifts to ethics. Ideally, ethics education in CS fosters the ability of students to understand and flexibly move between differing models of authority and the ethical models they imply.

Kristel Clayville is a lecturer in the computer science department at the University of Illinois Chicago where she also teaches in the medical school. She focuses on interprofessional education and communicating across disciplinary boundaries. She has clinical experience working in hospitals and teaches ethics in multiple professional contexts. She holds a PhD in religious ethics from the University of Chicago.

2.4 Stacy A. Doore

There are several complementary approaches to helping students critically examine ethical issues within their CS coursework. Embedded ethics lessons provide students with opportunities to practice discussing ethics and its relationship to responsible computing principles and practices within a range of CS topics. Dedicated ethics courses provide students the time for developing a more sophisticated set of reasoning skills and an ethical sensibility that can be applied to issues encountered in professional life. In either context, professors should be prepared to teach from a position of intellectual humility, examine their own values and assumptions, and model a professional identity concerned with the impact of their own work on society.

Stacy Doore is an Assistant Professor of Computer Science at Colby College in Waterville, Maine. She is a co-creator of the interdisciplinary Computing Ethics Narratives project and a member of the ACM Ethics and Computing Task Force. Her research and teaching focuses on emerging assistive technologies, multimodal interface design, applied ethics, and embedded responsible computing principles and practices. She holds a Ph.D. in Spatial Information Science and Engineering from the University of Maine.

2.5 Michael S. Kirkpatrick

Many CS faculty members have incorporated a learner-centered teaching philosophy into our courses, shifting from the "sage on the stage" model to the "guide on the side." Recognizing the societal impact of our rapidly evolving field, this shift is increasingly necessary to address the ethical issues our field has created. None of us have the answers for how to address these challenges and we must welcome the wide array of viewpoints created by a diverse set of stakeholders. Integrating ethical discussions into our discipline gives us an opportunity to model for our students the epistemic humility that is central to co-creating solutions to these problems.

Michael S. Kirkpatrick is an Associate Professor of Computer Science at James Madison University in Harrisonburg, VA. He was a member of the Executive Committee for the Code 2018 project that revised the ACM Code of Ethics and Professional Conduct and co-authored the Code booklet illustrating how to apply the Code. He is an officer for ACM Committee on Professional Ethics and Chair of the Ethics & Plagiarism committee of ACM Publications Board.

2.6 Michael Goldweber

When infusing ethical considerations into technical CS courses, one needs to relinquish the notion that there is a single correct answer. From an authority perspective, this helps, since the instructor no longer is the holder of said correct answer/truth. Instead, the goal is to expose students to different frameworks from moral philosophy and ask students to evaluate a given question using a given framework, or set of frameworks. This shift allows the notion of authority to migrate from one who holds the correct answer/truth, to that of the arbiter of what constitutes a well constructed argument.

Michael (Mikey) Goldweber is a Professor of Computer Science at Denison University in Granville, OH. Mikey is a past Chair of ACM SIGCAS and current member of its Executive Committee. MIkey is also one of the founders of the Computing for Social Good in CS Education movement. Finally, he was a member of the CS2023 Steering Committee where he served on the Society, Ethics, and Professionalism subcommittee.

Interviewing the Teaching Faculty Hiring Process

Geoffrey Challen
challen@illinois.edu
University of Illinois
Urbana, IL, USA

Victoria Dean
vdean@olin.edu
Olin College of Engineering
Needham, MA, USA

Nate Derbinsky
n.derbinsky@northeastern.edu
Northeastern University
Boston, MA, USA

Matt X. Wang
mxw@cs.washington.edu
University of Washington
Seattle, WA, USA

Jacqueline Smith (Moderator)
jsmith@cs.toronto.edu
University of Toronto
Toronto, ON, Canada

ABSTRACT

As teaching-focused positions proliferate and university teaching careers become more professionalized, there is growing attention being paid to how teaching faculty are created. However, how teaching faculty are hired also deserves scrutiny.

Teaching faculty hiring varies widely between institutions, raising questions of whether hiring processes are effectively identifying, evaluating, and recruiting qualified applicants, and which approaches are most effective. Variation in application requirements and interview processes may also result in a higher workload for teaching faculty candidates when compared to peers applying for other types of faculty positions.

This panel brings together faculty with significant teaching faculty hiring experience and new teaching faculty who were very recently on the job market. Together we'll discuss what is and is not working in teaching faculty hiring, and how we might improve the process for both institutions and candidates. We anticipate engaging with those involved with hiring teaching faculty, as well as current and future teaching faculty candidates.

CCS CONCEPTS

• **Social and professional topics → Computer science education**.

KEYWORDS

Teaching, Teaching track faculty, Computer Science Education

ACM Reference Format:
Geoffrey Challen, Victoria Dean, Nate Derbinsky, Matt X. Wang, and Jacqueline Smith (Moderator). 2024. Interviewing the Teaching Faculty Hiring Process. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3631664

1 SUMMARY

An increased focus on teaching excellence and the growing professionalization of post-secondary teaching has led many computer science departments to scale up their hiring of teaching-focused faculty. The nature of teaching faculty positions varies widely between institutions. Growing attention is being paid to the promotional expectations, workload, contract length, departmental rights, and other aspects of these roles [1]. There are regular conversations at the SIGCSE Technical Symposium about the nature of the roles [2] and who is qualified to take them [3]. However there seems to be limited discussion how teaching faculty candidates are recruited, interviewed, evaluated, and hired.

In our experience, hiring processes for teaching faculty vary more between institutions than "research+teaching" faculty processes. For example, many institutions require some kind of teaching demonstration, but the topic may be given by the institution, or chosen by the candidate; and it may be given to a real class, or to a group of faculty, who may pretend to be students. Some institutions also require a research presentation.

Our panel has four panelists and one moderator. Two panelists are senior faculty with experience hiring teaching faculty at separate institutions. Two panelists are junior faculty who interviewed for teaching faculty positions last year, and who have different backgrounds and different goals. The moderator is new to leading teaching faculty hiring, and is looking to improve the hiring process and candidate experience at their institution. Panelists will use their shared experience and perspectives from both sides of the hiring process to explore what's working and what's not, and to examine differences in the interview process across institutions.

The intended audience includes anyone interested in identifying and recruiting excellent and diverse teaching faculty, as well as potential candidates for teaching faculty positions. Due to the composition of our panel, the discussion will focus on hiring junior teaching faculty applicants at R1 universities. However, we anticipate that the discussion will encompass topics relevant to other kinds of institutions and to hiring more senior candidates.

2 PANEL STRUCTURE

All panelists will present in-person at SIGCSE TS 2024.

If this panel is selected to be livestreamed, a link will be provided for attendees to ask questions remotely and the moderator will alternate between in-person and online questions.

The panel will have two rounds of questions where panelists will share background on themselves and their institutions, and their experiences with teaching faculty hiring (e.g., the hiring process for senior panelists, and experience on the job market for junior

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3631664

panelists). Each panelist will have 5 minutes per round, for a total of 40 minutes. There will be 35 minutes for Q&A with the audience.

3 POSITION STATEMENTS

3.1 Geoffrey Challen

Geoffrey Challen is an Associate Teaching Professor in the Department of Computer Science at the University of Illinois. He has participated in teaching faculty hiring for several years. He was the first teaching faculty to chair the search at Illinois, and designed and implemented several changes to the hiring process.

The University of Illinois is a public R1 U.S. land-grant institution. Around 20 of 120 computer science faculty hold teaching-focused positions. Most teaching-track faculty at Illinois hold professorial titles, but other instructional titles are available, including ones that do not require a Ph.D. Teaching faculty at Illinois are not eligible for tenure, but are provided with multi-year contracts which are renewed annually. They vote on everything except tenure cases, and serve in department and college leadership roles.

Teaching faculty hiring at Illinois runs separately from research faculty hiring. Recently, teaching faculty candidates have been asked to submit a short prerecorded teaching demonstration, and give a interview talk on their approach to effective pedagogy in lieu of a live teaching demonstration.

3.2 Victoria Dean

Victoria Dean is an Assistant Professor of Computer Science at Olin College of Engineering, a small, private primarily undergraduate institution. She completed her PhD in Robotics at Carnegie Mellon University in 2023.

During the 2022–2023 job market cycle, she applied for teaching faculty positions at R1 universities as well as "research and teaching" faculty positions at primarily undergraduate institutions. She did onsite interviews at 8 primarily undergraduate institutions and 11 R1 teaching faculty positions. She also interacted closely with over a dozen candidates on the teaching faculty market in the same cycle. She can speak to the range of hiring practices across many institutions, including required application materials, teaching demonstration formats, and offer timelines. From her experiences and the experiences of her peers on the market, she can share insights about what candidates notice in the process and make suggestions for how hiring can be more candidate-friendly.

3.3 Nate Derbinsky

Nate Derbinsky is a Teaching Professor and the Associate Dean of Teaching Faculty in the Khoury College of Computer Sciences at Northeastern University. He has been involved in hiring dozens of (computing) faculty at multiple institutions in numerous countries.

Northeastern University is a private R1 institution with 13 campuses in the U.S., U.K., and Canada. Teaching-track faculty (called the Teaching Faculty) have professorial titles (Assistant / Associate / [Full] Teaching Professor), are classified as non-tenure-track (with renewable appointments, where length is based upon rank), and are included within a "One Faculty" model of shared governance (including full voting rights, eligibility for leadership, etc). At time of submission, Khoury has nearly 100 full-time teaching faculty at 8 campuses (as well as several that work remotely, teaching online courses). Most members of the Teaching Faculty hold a doctorate or equivalent terminal degree in their field; exceptions must be approved by the Provost in advance.

Teaching Faculty recruiting has many similarities with tenure-track recruiting, but operates on a rolling basis, potentially spans multiple campuses, and involves research presentations only if candidates express an interest in a research-active workload (teaching demonstrations are mandatory).

3.4 Matt Wang

Matt Wang is an Assistant Teaching Professor in the Paul G. Allen School of Computer Science at the University of Washington. He received his M.S. in Computer Science from the University of California, Los Angeles in 2023.

He underwent a narrow search in the 2022–2023 job market, applying to and doing onsite interviews for just two teaching-focused positions at R1 universities. He looks to provide insight from the unorthodox parts of his background. Prior to outreach from a current faculty member at the University of Washington, he was unaware of permanent teaching roles without doctoral degree requirements. Subsequently, applying without a PhD changed his approach to evaluating potential roles. He focused on materials outside of typical instructor-of-record experience; in his case, largely an emphasis on K-12 education, institutional advocacy, and open-source software development. He can suggest strategies for broadening the applicant pool in future teaching-track job searches, with a particular focus on applicants with less traditional teaching experiences.

3.5 Jacqueline Smith (Moderator)

Jacqueline Smith is an Associate Professor, Teaching Stream at the University of Toronto. She will be playing a larger role in teaching faculty recruiting this year, and is actively seeking to improve the hiring process.

The University of Toronto is a public R1 institution in Canada. Teaching faculty at the University of Toronto have professorial titles (Assistant/Associate/[Full] Professor, Teaching Stream), are eligible for tenure-equivalent "continuing status", and participate as full voting members of the department. Teaching Stream recruiting roughly parallels the "research+teaching" faculty recruiting process, requiring two teaching demonstrations instead of a research talk.

REFERENCES

[1] Penny Rheingans, Dan Grossman, and Jeff Forbes. 2018. Laying a Foundation: Best Practices for Engaging Teaching Faculty in Research Computing Departments. https://cra.org/teaching-faculty/ Accessed: Aug 16, 2023.

[2] Laney Strange, Melinda McDaniel, and Olga Glebova. 2023. Teaching Track Faculty in Computer Science. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1254. https://doi.org/10.1145/3545947.3573356

[3] Kendra Walther, Adam Blank, Michael Ball, and Suraj Rampure. 2023. A New Class of Teaching-Track Faculty: No Ph.D. Required. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1188–1189. https://doi.org/10.1145/3545947.3569608

Computing in Data Science or Data in Computer Science?

Exploring the Relationship between Data Science and Computer Science in K-12 Education

Zarek Drozda (Moderator)
The University of Chicago
Chicago, IL, USA
zarekd20@uchicago.edu

Justice Toshiba Walker
The University of Texas, El Paso
El Paso, TX, USA
jtwalker@utep.edu

Kathi Fisler
Brown University
Providence, RI, USA
kfisler@brown.edu

David Weintrop
University of Maryland
College Park, MD, USA
weintrop@umd.edu

ABSTRACT

Data literacy is quickly becoming an essential topic for all students to learn in order to succeed in an increasingly data-driven world. Foundational data literacy skills currently live in a number of subjects across K-12 (e.g., data collection and analysis in science classes, statistical calculations in mathematics/statistics, data visualization and communication in civics/social studies), however, a growing number of schools and districts are introducing stand-alone data science (DS) courses. Given the centrality of computing and programming in the contemporary practice of DS, many of these courses include topics historically reserved for computer science (CS) classes. Further, many CS courses include dedicated time for DS topics (e.g., AP Computer Science Principles' unit on Data). In many ways, DS educators and CS educators are working towards the same ends in complementary ways. However, at other times, the two disciplines are in tension, especially given the scarcity of time in K-12 student schedules for non-core subjects. This panel will explore what DS education and CS education can learn from each other, how each can contribute and advance the goals of the other, and how these two intertwined disciplines can productively live alongside each other in K-12 settings.

CCS CONCEPTS

• Social and professional topics → Computing education.

KEYWORDS

Data Science Education, Computer Science Education, K-12

ACM Reference format:

Zarek Drozda, Justice Toshiba Walker, Kathi Fisler and David Weintrop. 2024. Computing in Data Science or Data in Computer Science? Exploring the relationship between Data Science and Computer Science in K-12 Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V.1, March 20-23, 2-24, Portland, OR, USA. ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/3626253.3631654

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3631654
https://doi.org/10.1145/1234567890

1 PANEL SUMMARY

We are living in an increasingly data-driven world. At the heart of the data revolution are technologies and ideas drawn from computer science (CS). Despite this fact, contemporary data science (DS) concepts and practices are rarely incorporated into K-12 CS education. This is in part due to the relative recency of the ascendance of DS but also due to questions of disciplinary boundaries and the challenge of the rapidly expanding scope of CS. As a result, the emerging K-12 DS education movement [3, 4] is happening in parallel to recent K-12 CS education efforts [1, 2].

This panel will explore the relationship between the current state of K-12 CS education and the emerging K-12 DS education movement. While there are reasons for DS and CS to be taught independently there are also potentially powerful intellectual and practical reasons to integrate the two. Given K-12 education's goal of preparing youth to be informed citizens and recognizing the growing importance of both CS and DS, bringing these two fields together may help prepare youth to succeed in a computational and data-driven world.

The panel will start with an overview of the current state of K-12 DS education followed by short remarks from the moderator and panelists on their work at the intersection of DS and CS education in K-12 contexts (10 minutes each). The audience will be invited to discuss in small groups to generate and refine questions for the panel (5 minutes). The remainder of the session will be a moderated discussion weaving together audience-generated questions with discussion topics chosen by the moderator (30 minutes).

2 POSITION STATEMENTS

2.1 Zarek Drzoda (Moderator)

K-12 DS is growing rapidly: from 2019 to 2023, the Data Science 4 Everyone community grew from ~200 to 3,000 community members, from 1 to 17 states, and now includes over

70 K-12 programs. Recent national research convenings for DS education facilitated by the Institute of Education Sciences (2021), the National Academies of Sciences, Engineering & Medicine (2022), and the National Science Foundation (2023) have helped to further advance the goals of K-12 DS education. These efforts were made possible by the research and community infrastructure for K-12 CS education and concurrent initiatives to diversify K-12 mathematics with a greater emphasis on statistics, data, and technology. Critical decisions lie ahead for K-12 DS, including: Where in the K-12 schedule DS "fits"? How to best capture public and policymaker interest in DS, AI, and emerging technologies? and How to operationalize DS in K-12 education?

Zarek Drozda is the Director of Data Science 4 Everyone, a national initiative and coalition based at the University of Chicago. Zarek has worked at the intersection of applied research, data, and policy, including work at the U.S. Department of Education, Data Lead for the U.S. National COVID Response, and as an Analyst at the Center for RISC.

2.2 Kathi Fisler

Teaching DS and CS together, in an interwoven fashion, can address challenges that each area faces independently. From a content perspective, bringing DS into CS creates compelling real-world scenarios for basic programming tasks while also enabling discussions of ethical and responsible computing. Bringing CS into DS provides students with agency in data analysis and provides concrete ways to explain what it means to test computed results for accuracy. From a policy perspective, CS already has national standards with mandates in some states; including DS in computing makes it easier to deploy DS initiatives. Embracing the connections between CS, DS, and mathematics offers a way to increase the footprint of both CS and DS in K-12 courses without the need to revamp requirements.

Kathi Fisler is a Research Professor and co-Director of Undergraduate Studies in Computer Science at Brown University. For the last 5 years, Kathi has been developing and teaching an introductory CS course that integrates DS and computer science at Brown. Kathi exchanges ideas between this work and her long-standing K-12 work with Bootstrap, a program for K-12 teachers who wish to integrate computing and/or DS into their courses.

2.3 Justice Toshiba Walker

Situating DS in CS continues to draw attention as a viable frame for productive and agentic learning—given the potential of computing as a tool for accessing, processing, and analyzing data at scale. While research and education efforts show promise there are persistent concerns about the restrictive nature of activities that involve pre-generated data sources. During this panel, I will share a perspective we call 'Sandbox Data Science'—an approach that centers learners as agents in DS as CS— along with lessons learned from our work on the co-design of the Coding Like a Data Miner project.

Justice Toshiba Walker is a Learning Scientist and Assistant Professor of STEM Education at the University of Texas at El Paso, the nation's leading R1 Hispanic-Serving Institution, where he leads the ABC Learning Lab and its emphasis on learning technologies that leverage next generation bio and computing tools. His research examines how youth respond to learning paradigms that emphasize cultural relevance, epistemological agency and critical literacies.

2.4 David Weintrop

Today's youth are growing up surrounded by data. However, the ways youth experiences are shaped by data and the data youth create and often not fully transparent to the youth themselves. This lack of awareness and visibility can be problematic due to bias in how data is collected and used. At the same time, the data youth create and consume can serve as a compelling context for DS and CS learning in a way that resonates with their interests, identities, and experiences. As part of this panel, I will present emerging findings from the API Can Code project that situates DS and CS in the lived experience of today's high school students.

David Weintrop is an Associate Professor in the College of Education and the College of Information Studies at the University of Maryland. His research focuses on the design, implementation, and evaluation of effective, engaging, and equitable computational learning experiences.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grants No. (#2137708, #2141655). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Code.org and ECEP Alliance 2022. *2022 State of Computer Science Education: Understanding Our National Imperative.*

[2] Google & Gallup 2020. *Current Perspectives and Continuing Challenges in Computer Science Education in U.S. K-12 Schools.*

[3] LaMar, T. and Boaler, J. 2021. The importance and emergence of K-12 data science. *Phi Delta Kappan.* 103, 1 (Sep. 2021), 49–53.

[4] National Academies of Sciences, Engineering, and Medicine 2023. *Foundations of Data Science for Students in Grades K-12: Proc. of a Workshop.* National Academies Press.

Broadening Participation in Computing Education: Advancing LGBTQIA+ Voices

Wendy M. DuBow
dubow@colorado.edu
University of Colorado-Boulder
Boulder CO

Stephanie Jones
stephanie.jones@u.northwestern.edu
Northwestern University
Evanston IL

Stacey Sexton
ssexton@sagefoxgroup.com
SageFox Consulting Group
Amherst MA

Sri Yash Tadimalla
stadimal@uncc.edu
University of North Carolina
Charlotte NC

ABSTRACT

Broadening participation in computing (BPC) has been a key focus of the National Science Foundation (NSF) for over two decades. Its aim is to support students and faculty from historically underrepresented groups, including women, people with disabilities, and certain racial and ethnic groups. Within these communities, the diverse range of gender and sexual identities remains overlooked in computing education research (CER). To address this invisibility, this panel will discuss the benefits of integrating LGBTQIA+ perspectives. The moderator will provide context, define relevant terms, and set ground rules for discussion. The panelists will offer insights from a variety of perspectives, including: a discussion of the policy landscape impacting LGBTQIA+ students and the importance of incorporating their perspectives as researchers and participants; the erasure of queer history in computer science and advocate for LGBTQIA+ inclusion, considering the humanitarian calling for the field and CS educators in our tech-driven world; the current resistance to supporting LGBTQIA+ scholarship in computing, advocating for an inclusive approach; and finally, the experiences of marginalized individuals in CS education and ways to support them, emphasizing inclusivity through storytelling and personal narratives. The panel aims to increase visibility, understanding, and collaboration between the computing education research community and LGBTQIA+ individuals. By acknowledging and integrating diverse perspectives, we can begin to create a more inclusive, equitable computing landscape.

ACM Reference Format:
Wendy M. DuBow, Stephanie Jones, Stacey Sexton, and Sri Yash Tadimalla. 2024. Broadening Participation in Computing Education: Advancing LGBTQIA+ Voices. In *Proceedings of SIGCSE '24.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3631661

1 SUMMARY

This panel has as its primary goal to raise awareness of, and increase knowledge about, issues important to computing education and research (CER). Because issues related to the LGBTQIA+ communities in the realms of CER have not been broached formally in panels or papers at SIGCSE, or other important CER venues, we aim to begin a conversation that we hope will lead to many more. We request a hybrid session for the sake of inclusivity, so that the many people in this community who are unable to travel due to safety, health, or financial concerns still have an opportunity to be involved. LGBTQIA+ persons exist in CER, classrooms, and the technical workforce; thus, it is critical to acknowledge, respect, and integrate them. Increased diversity has positive benefits at the individual, institutional, and societal levels [1]. Inclusion of diverse perspectives has positive impacts on innovation and national capacity [3]; additionally multicultural classrooms have long-term impacts on the attitudes and beliefs of the students in them [2]. Beyond strengthening outcomes related to the learning and workforce competency goals that we hold for computing and engineering, there is an ethical imperative to acknowledge and support those who are being marginalized and oppressed. We work within this imperative. The intended audience is two-fold: Any SIGCSE attendees who identify as LGBTQIA+ and allies who want to learn more about what individuals and research from these LGBTQIA+ communities can offer the CER field.

2 PANEL STRUCTURE

There will be four panelists and one moderator. The four panelists have lived experience relative to the issues being discussed, and the Moderator has lived experience as an ally. They have also all been involved in CER. The Moderator will provide context, definitions of terms, and set ground rules for an accountable and learning space since the identities we will be sharing and discussing are under attack. One panelist, and we believe, many participants will be online in our proposed hybrid format. Each panelist will give a brief starter on their selected topic and share their positionality. We will make use of slides and handouts. After the brief starters, we will invite audience members from the LGBTQIA+ community to add their knowledge to the conversation. Subsequently, the moderator will solicit audience questions. Our goal is to provoke thought and discussion and provide resources for further learning. Per the schedule below, we are limiting our time per speaker in order to have broad representation on the panel and also recognize and incorporate audience expertise.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–24, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-0424-6/24/03.
https://doi.org/10.1145/3626253.3631661

3 POSITION STATEMENTS

3.1 Wendy DuBow (Moderator)

Dr. Wendy DuBow holds a director position on the social science team at the National Center for Women & Information Technology (NCWIT) and is an affiliate faculty member in Women and Gender Studies at the University of Colorado-Boulder. At NCWIT, she conducts mixed methods social science research and directs evaluation. She has been PI, co-PI, and senior personnel on NSF and other computing-related grants since 2009. DuBow is a cis-gender woman, second-generation in a refugee family, and parent of two adult children, one of whom is a member of the LGBTQIA+ community. She strives to activate her allyship to individuals in the LGBTQIA+ community through her research as well as personal and professional interactions.

3.2 Stacey Sexton

Stacey Sexton is a researcher and evaluator in CER with 10 years of experience. They are a member of the LGBTQIA+ community and bring their lived experience in addition to empirical knowledge. Sexton will discuss the policy landscape that we are living within and connect it with the lived experiences of the students in CS. They will note that there are many other concerns that LGBTQIA+ people are facing beyond their experiences within CS spaces. They will highlight the importance of situating LGBTQIA+ researchers as also experiencing forces of marginalization and calling for solidarity. Sexton, along with panelist Menier, facilitated a workshop for Trangender and Nonbinary researchers in CS and Engineering in which we collectively developed an initial research, policy and practice agenda that explored intersectional identities for these students as they move through the CS ecosystem.

3.3 Sri Yash Tadimalla

Sri Yash Tadimalla is a Ph.D. student in the College of Computing and Informatics at UNC Charlotte, where he is pursuing an interdisciplinary degree in Computer Science and Sociology. There he assists with various NSF research projects under the Center for Education Innovation (CEI) Lab and the Human-Centered Computing (HCC) Lab. His doctoral work focuses on Using an intersectional positionality approach on student experience to understand the lack of diversity in the field of AI and Computer Science. He is the General-Secretary for the World Student Platform for Engineering Education and Development (SPEED) and a member of the United Nations Major Group for Children and Youth. He is a first-generation South-east Indian immigrant and graduate student. He identifies as a cis-man and uses the pronouns He/Him/His. Tadimalla will talk about LGBTQIA+ students' sense of belonging; share personal motivations/barriers while pursuing CS; and what does advocacy look like at state/federal/global levels. He will address the humanitarian calling for our field and CS educators and how "leave no one behind" provides a framework for LGBTQIA+ inclusion in CS.

Elapsed Time	Topic
:00-:10	Moderator: Introduction and norms
:11-:15	Panelist 1
:16-:20	Panelist 2
:21-:25	Panelist 3
:26-:30	Panelist 4
:31-:35	Audience: Individual reflection
:36-:60	Audience comments and questions
:61-:75	All: Panelist responses & final comments

Table 1: Proposed Use of Time

3.4 Stephanie T. Jones

Stephanie T. Jones is a Ph.D. Candidate in Computer Science and Learning Sciences PhD program at Northwestern University. Jones's interests span context and intellectual communities from Black Studies, Education, Computing, and Abolition. Their research interests include intergenerational learning opportunities and the relationships between anti-Blackness, Black Life, learning, and STEM. She is motivated by her own experiences as the only Black graduate of the 2018 class at Villanova University in both Computer and Electrical Engineering. She has been supported by the National GEM Consortium, the National Science Foundation, and the National Society of Black Engineers. Jones is a non-binary Black femme from the Philadelphia area and uses they/she pronouns. Jones will explore Computer Science Education at the margins. They engage in storytelling to describe how people of marginalized backgrounds are experiencing CS in different ways and how we can support them. They will facilitate a space of reflection for the group before we enter our question and answer segment.

ACKNOWLEDGMENTS

Funding was provided in part by National Science Foundation CISE 2216561.

REFERENCES

[1] Jeffrey F Milem. 2003. *The educational benefits of diversity: Evidence from multiple sectors.* Standford University Press, 126–169.
[2] Heather Smith, Ray Parr, Raven Woods, Brooke Bauer, and Tim Abraham. 2010. Five years after graduation: Undergraduate cross-group friendships and multicultural curriculum predict current attitudes and activities. *Journal of College Student Development* 51, 4 (2010), 385–402.
[3] Trung V Vu. 2022. Linking LGBT inclusion and national innovative capacity. *Social Indicators Research* 159, 1 (2022), 191–214.

Technology We Can't Live Without!

Daniel Garcia (Moderator)
UC Berkeley
Berkeley, CA, USA
ddgarcia@berkeley.edu

Delnavaz Dastur
Stratford San Jose Middle School
San Jose, CA, USA
ddastur@stratfordschools.com

Kimberly Hermans
Irvine Unified School District
Irvine, CA, USA
kahermans@gmail.com

Nery Chapeton-Lamas
MiraCosta College
Oceanside, CA, USA
nchapetonLamas@miracosta.edu

Ariel Ortiz
Tecnologico de Monterrey
Atizapán de Zaragoza, Edo. Méx.
Mexico
ariel.ortiz@tec.mx

ABSTRACT

This panel is the evolution of seven years of *Technology that Educators of Computing Hail (TECH)* Birds of a Feather sessions, grew into popular panels [7–12], and a regular column in ACM Inroads [1, 21]. Leveraging our experiences of teaching during the pandemic, many of us have incorporated technology we were forced to rely on during our time online, making for an even *better* experience for our students. This forum will provide a chance for a seasoned middle school, high school, community college, and university educator to showcase the technologies they can't live without, what problems they solve, and how to use them.

CCS CONCEPTS

• **Social and professional topics → Computer science education**.

KEYWORDS

Computer science education; technology and teaching; tools

ACM Reference Format:
Daniel Garcia (Moderator), Delnavaz Dastur, Kimberly Hermans, Nery Chapeton-Lamas, and Ariel Ortiz. 2024. Technology We Can't Live Without!. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3631662

1 SUMMARY

The pace of technology for use in computing education is staggering; in recent years, *Zoom*, *Piazza*, *Gradescope*, etc. have completely transformed our teaching. We hope this forum will allow the presenters to dive deeply into the common use cases of their indispensable technologies, share any "gotchas" they've uncovered, and explain how others can adopt them.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3631662

2 PANEL STRUCTURE

After a brief introduction of the panel format and background, our presenters (ala veteran concert musicians) will take ten minutes each to give live demonstrations and reveal the configuration options required to make their technology "sing". When not presenting, the presenters will be answering questions in the chat; one of the benefits of hybrid conferences and Pathable!

We will support in-person and remote presenters; the latter will pre-record their demo, which will be shown by the moderator during their turn. The highlight of the panel is when the audience is invited (in the remaining 30 minutes) to interact, join the discussion, and share their favorite "can't live without" technologies as well. The moderator will alternate between selecting raised hands of in-person participants and remote attendees.

The overall format has worked well for many years [7–12] with a moderator and four presenters, bringing their perspectives.

3 DELNAVAZ DASTUR

I use **Pear Deck for Google slides** [3] to allow students to give me feedback during lectures, making it very interactive. I can take a poll, ask quick questions to get an understanding of students' thoughts about the material, or have students work at their own pace and complete questions on each slide. I can view student responses in real time on my laptop as they type and can give immediate feedback. Pear Deck has slide templates for different subjects; I use them for critical thinking as well as reflection questions at the end of a lesson. **Gimkit** [13] is invaluable to reinforce concepts; it has fun games where a class is either split into teams or works as one team to win the game. Each student has to answer questions correctly that the teacher has created or selected in order to win the game. I utilize **Screencastify** [18] to record short instructional videos, and **edpuzzle** [6] to assign videos for my students to watch. I can add short questions in the middle of videos to check for understanding of concepts, as well as trim or cut parts of the video. I can see how many times the videos were watched, played back, and which parts in particular were played multiple times.

Delnavaz Dastur is a seasoned Middle School CS Teacher at Stratford San Jose Middle School.

4 KIMBERLY HERMANS

Pair programming and interactive coding demonstrations are an integral part of my classroom, so I use **CodeTogether** [2] daily.

CodeTogether is a plugin that allows you to live share your IDE and coding sessions. I have a paid subscription (around fifty dollars a year) that allows me to share my IDE with my class, who can view and edit through a web browser. It allows me to call on students to contribute code and it allows students to copy and paste demonstration code for their personal use. Students can sign up for a free account that allows them to share their IDE with up to 3 other students.

Another tool I use in the classroom is **Thunkable** [22], a mobile app builder using a block based language that allows for in-browser testing. Students can also download the Thunkable app on a mobile device that loads and runs their projects. Other tools I use to help build community in my classroom include **Google's Jamboard** [15] and **Doodle** [5]. I use Jamboard as a way to get to know my students by asking fun questions, and students respond with pictures or text posted on a communal bulletin board. I also use Doodle in my class to have students sign up for things like presentations and snacks.

Kimberly Hermans is a veteran high school teacher, and teaches computer science in the Irvine Unified School District.

5 NERY CHAPETON-LAMAS

I use **Replit Teams for Education** for classroom management, and for mob / pair programming (especially online/zoom classes). **Kahoot** is great for reviewing in a flipped class, making for easy debriefs between questions or "what if" program scenarios. Like Kimberly, I also use **Google Jamboard** [15], which works well for discussions on culturally responsive contextual topics to setup for lab. **Google Sheets** is invaluable (since Canvas stinks for *specification grading*), I have a custom sheet with automatic tallying for end of unit/semester grades. Our clubs use **Discord**, so using it as our online classroom space to build community also helps pipe students into our *Women in CS* club and *EChALE* (Empowering Chicanx And Latinx in Exploring) STEM club. I've found **Google Voice** invaluable; it allows me to give students a cell number they can text, another option for communication that some prefer over Discord, Email, Canvas, etc. Finally, **Slack** is awesome for organizing our department and peer tutors, as it makes file sharing, communicating, and collaborating way easier than email.

Nery is a full-time tenured instructional faculty in Computer Science at MiraCosta Community College in Oceanside, CA.

6 ARIEL ORTIZ

Using a web IDE on a browser is a very convenient way of making sure all students in our classes have access to the exact same programming environment through their own personal laptop computers. If we are using Python in our classes, **Google's Colaboratory** [16] allows creating **Jupyter notebooks** [17], where we can mix code, text, LaTeX formulas, and images. If what we need is a Linux environment, there are two options worth considering: **Cloud9** [20] through the **AWS Academy** [19] and **GitPod** [14]. Both these IDEs can be employed to program in any language that works in a text-based Linux system, including: C, C++, C#, Java, JavaScript, Python, Ruby, Go, and Rust. Cloud9 and GitPod allow us to install `valgrind` [4], which is absolutely necessary when teaching memory management in C/C++, in order to diagnose and remedy all kinds of memory-related issues, such as memory leaks and array index out of bounds errors. All these tools are either open source software or free for academic use.

On another note, now that we're back to in-person classes, don't you miss having a second monitor like when you were teaching remotely from home during the pandemic? If you have a Mac laptop you can easily configure an iPad to work as an extended monitor while simultaneously mirroring your laptop's screen into the classroom's projector.

Ariel Ortiz has been a computer science undergraduate educator at Tecnológico de Monterrey in Mexico for more that three decades.

REFERENCES

[1] Leslie Aaronson. 2016. TECHNOLOGY THAT EDUCATORS OF COMPUTING HAIL (TECH); Create a Digital Footprint of CS Work Using Digital Portfolios. *ACM Inroads* 7, 4 (Nov. 2016), 22–23. https://doi.org/10.1145/3009958

[2] CodeTogether. 2022. CodeTogether. https://www.codetogether.com/ [Online; accessed 2022-08-16].

[3] Pear Deck. 2022. Pear Deck. https://peardeck.com/ [Online; accessed 2022-08-16].

[4] Valgrind Developers. 2022. Valgrind Home. https://valgrind.org/ [Online; accessed 2022-08-16].

[5] Doodle. 2022. Doodle. https://doodle.com/ [Online; accessed 2022-08-16].

[6] edpuzzle. 2022. edpuzzle. https://edpuzzle.com/ [Online; accessed 2022-08-16].

[7] Ria Galanos, Michael Ball, John Dougherty, Joe Hummel, and David J. Malan. 2018. Technology We Can't Live Without!, Revisited. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 1043–1044. https://doi.org/10.1145/3159450.3159629

[8] Ria Galanos, Whitaker Brand, Sumukh Sridhara, Mike Zamansky, and Evelyn Zayas. 2017. Technology We Can't Live Without! Revisited. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 659–660. https://doi.org/10.1145/3017680.3017691

[9] Dan Garcia, Zelda Allison, Abigail Joseph, David J. Malan, and Kristin Stephens-Martinez. 2022. Technology We Can't Live Without! (COVID-19 Edition). In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2* (Providence, RI, USA) *(SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 1043–1044. https://doi.org/10.1145/3478432.3499221

[10] Dan Garcia, Tiffany Barnes, Art Lopez, Chinma Uche, and Jill Westerlund. 2021. *Technology We Can't Live Without!, Revisited.* Association for Computing Machinery, New York, NY, USA, 138–139. https://doi.org/10.1145/3408877.3432571

[11] Daniel D. Garcia, Leslie Aaronson, Shawn Kenner, Colleen Lewis, and Susan Rodger. 2016. Technology We Can't Live Without!, Revisited. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 236–237. https://doi.org/10.1145/2839509.2844668

[12] Daniel D. Garcia, Eric Allatta, Manuel Pérez-Quiñones, and Jeff Solin. 2015. Technology We Can't Live Without!. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (Kansas City, Missouri, USA) *(SIGCSE '15)*. Association for Computing Machinery, New York, NY, USA, 597–598. https://doi.org/10.1145/2676723.2677336

[13] Gimkit. 2022. Gimkit. https://gimkit.com/ [Online; accessed 2022-08-16].

[14] Gitpod. 2022. Gitpod. https://www.gitpod.io/ [Online; accessed 2022-08-16].

[15] Google. 2022. Jamboard. https://jamboard.google.com/ [Online; accessed 2022-08-16].

[16] Google. 2022. Welcome to Colaboratory. https://colab.research.google.com/ [Online; accessed 2022-08-16].

[17] Project Jupyter. 2022. Jupyter Notebooks. https://jupyter.org/ [Online; accessed 2022-08-16].

[18] Screencastify. 2022. Screencastify. https://screencastify.com/ [Online; accessed 2022-08-16].

[19] Amazon Web Services. 2022. AWS Academy. https://aws.amazon.com/training/awsacademy/ [Online; accessed 2022-08-16].

[20] Amazon Web Services. 2022. AWS Cloud9. https://aws.amazon.com/cloud9/ [Online; accessed 2022-08-16].

[21] Jeff Solin. 2017. TECHNOLOGY THAT EDUCATORS OF COMPUTING HAIL (TECH); Using Cloud9, a Powerful Cloud-Based IDE in the Classroom. *ACM Inroads* 8, 1 (Feb. 2017), 29–30. https://doi.org/10.1145/3043953

[22] Thunkable. 2022. Thunkable. https://thunkable.com/ [Online; accessed 2022-08-16].

Enduring Lessons from 'Computer Science for All' for AI Education in Schools

Shuchi Grover
Looking Glass Ventures
Austin, TX, USA
shuchigrover@acm.org

Deborah Fields (Moderator)
Utah State University
Logan, UT, USA
deborah.fields@usu.edu

Yasmin Kafai
University of Pennsylvania
Philadelphia, PA, USA
kafai@upenn.edu

Shana White
Kapor Center
Oakland, CA, USA
shana.white@kaporcenter.org

Carla Strickland
UChicago STEM Education
Chicago, IL, USA
castrickland@uchicago.edu

ABSTRACT

Even as efforts to promote K-12 CS education forge ahead, there is a growing consensus that students must also be taught artificial intelligence (AI) and machine learning (ML) in order to be prepared for the fast-changing world powered by AI/ML. How can ensure that we leverage learnings from two decades of CS education research and practice, and build on successes while mitigating missteps? This panel invites researchers with deep expertise in 'CSForAll' efforts for a timely discussion and sharing of valuable lessons from CS education efforts about pedagogies, attention to equity, and teacher preparation that will also benefit K-12 AI education.

ACM Reference Format:
Shuchi Grover, Deborah Fields (Moderator), Yasmin Kafai, Shana White, and Carla Strickland. 2024. Enduring Lessons from 'Computer Science for All' for AI Education in Schools. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3631656

1 SUMMARY

The meteoric growth in the range and diversity of applications of ML in all industries and everyday products (with Large Language Models being the latest example) has fueled an unrelenting push to promote AI education in schools. As we aim for equitable teaching about the technologies powering AI, learners of all ages must also engage with issues of ethics and a critical examination of AI.

This new 'K-12 AI in Education' movement evokes a sense of déjà vu for those deeply involved in recent–and ongoing–CS For All (CSForAll) efforts. As "AI4K12" takes off, this panel serves as a timely reminder that we need not reinvent the wheel when it comes to designing for "AI learning for All", but instead leverage learnings from two decades of active work in K-12 CS education to ensure that we build on successes and mitigate failures as we enter what appears to be "K-12 CS 2.0". Given that AI is a subfield of CS (in that there can be no AI without computers), we must heed the many valuable lessons we have learned from the CSForAll movement related to classroom pedagogies, curricular strategies, equity and inclusion, skills definition, assessment, and teacher preparation [4].

This all-female panel brings together five researchers (including a moderator) who have been steeped in research and practice in K-12 CS education in the US and gained deep insights into computing education in schools. Each panelist will address a distinct but key learning from K-12 CS that we believe will collectively aid AI education efforts, and that also speak to lessons from early research in AI education in schools. These topics include culturally relevant and equity-focused pedagogies for inclusion, project-based and interest-driven constructionist learning, computational thinking (CT) and strategies for integrating AI into other subjects, and strategies for teacher preparation. In addition to speaking on their own work, panelists will also draw on pertinent lessons from the broader body of K-12 computing education research. Key takeaways will include: What are enduring lessons from CSForAll that can be leveraged as we design for AI education? How can we avoid reinventing the wheel and/or repeating missteps from earlier CS education efforts? What is the same and what is unique about AI education vis-a-vis CS education in K-12?

2 PANEL STRUCTURE

After an introduction to the session by the moderator (5 mins), each panelist will have seven minutes to introduce their position, highlighting lessons learned about a key aspect of CS education over the last two decades, and articulate key ideas for informing AI education research and practice. The moderator will then pose prepared questions to the panelists who will each have a chance to answer (approximately 20 minutes) followed by audience Q&A (20 mins). Candidate prepared questions include: **(1)** What have been the major breakthroughs in working toward equity in CS education–in curriculum, pedagogy, policy, or teacher education? What lessons learned could be applied to developing equity in AI education in K-12 grades? **(2)** Several tools and curricular activities have been developed for K-12 CS (e.g., platforms like Scratch and App Inventor, plugged and unplugged activities). What are some takeaways regarding designing for AI education? **(3)** One of the key aspects of K-12 CS education is teacher preparation and retention. What does this look like for AI education? **(4)** What do you see as similarities and differences between CS and AI education? Are

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3631656

there areas where lessons from CS education do not apply? **(5)** What lessons does the evolution of CSForAll have for K-12 AI education?

3 SHUCHI GROVER

A key breakthrough in K-12 CS and programming education was the recognition that CT [6] comprises problem-solving approaches that transcend programming environments. Over time, conceptualizations of CT have expanded to include situated and critical framings [7]. CT-related research has lessons we can take forward to AI education to not only help focus on skills needed to critically interrogate and develop AI apps but also use it as a lever for integration into other subjects. While CT 1.0 was rule-driven and emphasized algorithmic steps, 'CT for AI' is data-driven, emphasizing skills such as being able to understand and experiment with data [1, 4, 12]. As a K-12 CS education and AI education researcher, I will draw on my own work as well as broader research to discuss CT for AI, and integration of AI into various subjects (e.g. [5]).

4 DEBORAH FIELDS (MODERATOR)

I am a researcher, curriculum writer, and teacher educator focusing on the learning ecologies [8] of computer science education—i.e., intersections between students' computational project creations, curriculum development, curriculum implementation, teacher training, and the roles of parents, administrators, counselors, school systems, and policy. All of these aspects play a role in shaping what K-12 CS education looks like both locally and nationally. As moderator of this panel, I will frame the session in light of our CS education history and the ecology of learning—from researchers to teachers to students to policy makers—that we must draw on as we look toward AI education. During Q&A, I will elicit some of these different, connecting aspects of CS education across our panelists' expertise as we look ahead to developments in K-12 AI education.

5 YASMIN KAFAI

As a researcher working for three decades in CS education, primarily project-based, constructionist learning, I have been a part of its ebbs and flows. A noteworthy recent transformation within K-12 CS education has been the paradigm shift that positions learners as creators of genuine, real-world applications with personal relevance. These applications span a spectrum from digital games to narratives, seamlessly incorporating subject-specific content. This approach underscores the versatility of programming, highlighting its relevance across diverse activities, and extending beyond its conventional technical confines, while also compelling creators to thoughtfully consider how users will engage with the applications they design, and introducing a social dimension to their role. I will share valuable insights encompassing the dimensions of authenticity, relevance, integration, and community [3]. The critical importance of these dimensions is evident in the hundreds of millions of varied projects, shared by youth on popular programming platforms like Scratch, Minecraft, and Roblox.

6 SHANA WHITE

In the new push for AI in K-12 CS education, our central themes of this realm of computing should not just be centered around teaching AI. Instead, we must also push the understanding of ethics in AI, social justice implications of computing on society as well as specifically on historically marginalized communities [2], and the centering of equity and inclusivity in our teaching and learning about AI in K-12 classrooms [9]. AI cannot be treated in a vacuum, but instead we should afford all K-12 students and teachers ample opportunity to critically interrogate AI's utility, its dangers, and its vast implications on society at large. I bring my experience and expertise as Director of CS Equity Initiatives at the Kapor Center, a CS education researcher, and a former K-12 CS teacher to discuss these critical imperatives for K-12 CS and AI Education.

7 CARLA STRICKLAND

K-12 CS education has come to appreciate the crucial role that teachers play in the success of CSForAll efforts. Pre-existing pedagogical and content expertise of K–12 educators must be prioritized in the design and development of curriculum and professional learning (PL) experiences to prepare in-service teachers to teach CS. Integration provides pathways to not only find time for CS in an already full school day [11], but also to allow K-12 teachers, without specialized CS content knowledge, to leverage and apply equity-based pedagogies to create impactful learning experiences for ALL their students [10]. I will share lessons learned about in-service PL and curriculum development from my experience with elementary CS education at the intersection of research and practice.

REFERENCES

[1] Brian Broll and Shuchi Grover. 2023. Beyond Black-Boxes: Teaching Complex Machine Learning Ideas through Scaffolded Interactive Activities. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 15990–15998.

[2] MMG EARTH. n.d.. The Ai X Racial Justice Toolkit. https://www.mmg.earth/juneteenth-release-ai-x-racial-justice-toolkit.pdf.

[3] Deborah Fields and Yasmin Kafai. 2023. Supporting and Sustaining Equitable STEAM Activities in High School Classrooms: Understanding Computer Science Teachers' Needs and Practices When Implementing an E-Textiles Curriculum to Forge Connections across Communities. *Sustainability* 15, 11 (2023), 8468.

[4] Shuchi Grover. 2024. Teaching AI to K-12 Learners: Lessons, Issues, and Guidance. In *Proceedings of the 55th ACM SIGCSE conference.*

[5] Shuchi Grover, Brian Broll, and Derek Babb. 2023. Cybersecurity Education in the Age of AI: Integrating AI Learning into Cybersecurity High School Curricula. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1.* 980–986.

[6] Shuchi Grover and Roy Pea. 2018. Computational thinking: A competency whose time has come. *Computer science education: Perspectives on teaching and learning in school* 19, 1 (2018), 19–38.

[7] Yasmin B Kafai and Chris Proctor. 2022. A revaluation of computational thinking in K–12 education: Moving toward computational literacies. *Educational Researcher* 51, 2 (2022), 146–151.

[8] Nichole Pinkard. 2019. Freedom of movement: Defining, researching, and designing the components of a healthy learning ecosystem. *Human Development* 62, 1-2 (2019), 40–65.

[9] Allison Scott, Shana V. White, Kalisha Davis, Dinah Becton-Consuegra, Sonia Koshy, and Frieda McAlear. 2021. Culturally responsive-sustaining computer science education: A framework. Kapor Center.

[10] Carla Strickland, Andrea Ramírez-Salgado, Lauren Weisberg, LaToya Chandler, Jeanne Di Domenico, Elizabeth M Lehman, and Maya Israel. 2023. Designing an Equity-Centered Framework and Crosswalk for Integrated Elementary Computer Science Curriculum and Instruction. (2023).

[11] Carla Strickland, Kathryn M Rich, Donna Eatinger, Todd Lash, Andy Isaacs, Maya Israel, and Diana Franklin. 2021. Action Fractions: The design and pilot of an integrated math+ CS elementary curriculum based on learning trajectories. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education.* 1149–1155.

[12] Henriikka Vartiainen, Tapani Toivonen, Ilkka Jormanainen, Juho Kahila, Matti Tedre, and Teemu Valtonen. 2021. Machine learning for middle schoolers: Learning through data-driven design. *Intl Journ. of Child-Comp. Interaction* 29 (2021), 100281.

Re-making CS Departments for Generation CS

Kathleen J. Lehman (moderator)
katejlehman@ucla.edu
University of California, Los Angeles
Los Angeles, CA, USA

Carla Brodley
c.brodley@northeastern.edu
Northeastern University
Boston, MA, USA

Mark Guzdial
mjguz@umich.edu
University of Michigan
Ann Arbor, MI, USA

Paul Tymann
paul.tymann@gmail.com
Rochester Institute of Technology
Rochester, NY, USA

Aman Yadav
ayadav@msu.edu
Michigan State University
East Lansing, MI, USA

ABSTRACT

In response to the enrollment surge that started in many CS departments around 2006, the Computing Research Association published a report on "Generation CS", which named a pervasive theme in computing education: more and a greater diversity of students are seeking computing education, even if not as traditional CS majors. However, our curricula and departments have stayed much the same. We still mostly prepare students for software development jobs in the technology industry, while we rarely identify the damage that same industry has caused in our democratic societies. How do we do better? How do we change to meet the needs of a changing society? What strategies should we apply? We know that large-scale change will require structural shifts, but such shifts are likely to be slow and expensive, whereas smaller, "boots on the ground" initiatives can positively impact individuals but do little to change the systems that underlie the deeper-seated problems in computing. Navigating this paradox is imperative to our success as a field. Our panel will address the big questions about how to make structural changes in computing education in order to meet the greater needs of Generation CS.

ACM Reference Format:
Kathleen J. Lehman (moderator), Carla Brodley, Mark Guzdial, Paul Tymann, and Aman Yadav. 2024. Re-making CS Departments for Generation CS. In *Proceedings of Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.363165

1 SUMMARY

How do we change CS departments? We still mostly prepare students for software development jobs in the tech industry; we are rewarded for staying the course. The existing structures may have met the traditional needs of a CS department, but that leaves a lot of Generation CS students' needs unmet. There are more non-CS majors taking our classes, but the classes are not designed for them. There are more diverse students starting CS, but they're not being retained. What should we be providing to students who are not pursuing software development careers (e.g., computational scientists)? How do we prepare future citizens to question the role of computing in our society and to improve it? How do we make computer science truly interdisciplinary? How do we make sure that the K-12 students who are engaged by their "CS for All" experiences do not face higher education CS roadblocks from limits on enrollment and GPA caps?

Our panel will address the big questions about how to make structural changes in computing education in order to meet the greater needs of Generation CS. Can we make changes within our existing higher education systems, or do we need to tear down the walls and start over with something designed for diversity, inclusion, interdisciplinarity, and equity from the outset?

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.363165

2 PANEL STRUCTURE

Introduction (5 minutes): The moderator will provide context for the discussion, describing why evolving departmental practice is important and introduce the panelists.

Panelist Statements (5 minutes each): Each panelist will provide brief statements, offering their perspectives on how to best change CS departments to meet the needs of 'Generation CS'.

Discussion(40 minutes): The moderator will facilitate discussion of questions posed by the audience. We anticipate that we will collectively engage in questions at the heart of this dilemma: what efforts should be prioritized? How do we fund them? How do we evangelize others in computing education to our cause?

3 PANELISTS

Kathleen J. Lehman (Moderator): Dr. Lehman serves as the Associate Director for *Momentum*: Accelerating Equity in Computing and Technology, a research hub housed in the School of Education and Information Sciences at UCLA that conducts mixed-methods research on efforts to diversify computing and technology fields. Dr. Lehman started her career in academia working with student programs in STEM. After watching women and minoritized students struggle in these programs, ultimately leading many to switch to non-STEM majors, Dr. Lehman moved into research, where she would be better positioned to impact best practices and institutional change efforts. With Dr. Linda Sax at UCLA, she has lead multiple NSF-funded studies focused on recruiting and retaining women and racially/ethnically minoritized students to undergraduate computing majors, including work with the BRAID initiative and Center for Inclusive Computing. *Momentum* receives inquiries

from CS departments all over the world asking "what works" in terms of diversifying computing. This is a fraught question—there is not a magic bullet—our research suggests that most student-level change efforts operate as "thumbs on the scale" that may decrease or increase the odds of desirable outcomes, but no one program or initiative will remake the demographics of undergraduate computing.

Carla Brodley is a Professor in the Khoury College of Computer Sciences at Northeastern University where she co-leads the Center for Inclusive Computing (CIC) with Catherine Gill. She previously served as Dean of Khoury from 2014-2021, during which she made many curricular and policy changes to improve both the attraction and retention of students of all intersectional identities, including scaling interdisciplinary computing via combined majors. By 2021, over half of the 3000 computing undergraduates were pursuing a combined major. Students could select from 43 different options such as cybersecurity and criminal justice and sociology and data science. These changes directly impacted the diversity of computing Northeastern: the number of women majors grew from 19% to 32% and the number of students from races/ethnicities historically marginalized in tech grew from 11% to 19% [2].

Interdisciplinary computing degrees address the expanding need for computing knowledge and skills across all disciplines and enable universities to provide more relevant and diverse computing experiences for multi-talented students, who are often forced to select one discipline over another, at great detriment to their personal growth and future professional options/directions. Such degrees increase student employability because interdisciplinary students tend to be strong in computing and can synthesize computing across other non-computing disciplines (a skill traditional CS students often must develop independently). The CIC is currently working on a national demonstration project with eight large public universities to explore the cost of implementing interdisciplinary computing majors.

Mark Guzdial is a Professor in the Computer Science & Engineering Division at of the Electrical Engineering & Computer Science Department of the University of Michigan. He is the Director of the Program in Computing for the Arts and Sciences (PCAS) in the College of Literature, Science, and the Arts (LSA). At Georgia Tech, he led the development of an introductory course in computing for liberal arts, architecture/design, and business/management students that had a significantly higher retention rate than comparison classes and was adopted by other institutions with similar positive retention impacts [3].

Dr. Guzdial co-led a task force (with Dr. Gus Evrard) in LSA that asked the question, "What do LSA students need to know about computing?" They identified three themes of how computing is used in LSA, that are mostly not addressed in CS programs: Computing for *Discovery* (as in computational science), Computing for *Expression* (from social media to video games), and Computing for *Justice* (considering how computing systems exacerbate inequities, and might be designed to support democratic society). LSA is creating PCAS to meet these needs which are not typically met in CS programs.

Paul Tymann is a Program Director in the Division of Undergraduate Education in the Education and Human Resources Directorate (EHR/DUE) at the National Science Foundation. Prior to joining the NSF in January 2020, Tymann was a professor and the former Chair of the Computer Science Department at the Rochester Institute of Technology. Tymann has been involved in computer science education at the high school level and served as the co-Chair of AP Computer Science Development Committee from 2011 until 2015 and has served as the Chief Reader for the AP Computer Science Principles Exam from 2014 until 2020.

While Chair of the CS department at RIT, Tymann oversaw a successful redesign of the introductory programming sequence. After serving as Chair, Tymann was approached by the Dean of the College of Liberal Arts (COLA) at RIT to help incorporate computing into the various degree programs offered by COLA. The result was the development of a computing program for "non-developers." We all know that change is hard even in the best of circumstances, but his experience working in academia has led him to believe that change is impossible without having both bottom-up and top-down support. This statement is not all that surprising, but the question that it leads to is how do you garner the support you need to create real and lasting change?

Aman Yadav is the Lappan-Phillips Professor of Computing Education in the College of Education and College of Natural Science at Michigan State University. Dr. Yadav's research interests focus on preparing primary and secondary teachers to teach computational thinking and computer science. Recently, he has been working on justice-focused computer science education and how to prepare teachers to center their students' lived experiences, bring criticality, and help CS students understand their role as citizens [4, 5]. Research is also suggesting that CS undergraduate majors are the least likely of all disciplines to believe they are working to foster justice in the world. In order to dismantle the role of CS in reproducing White Supremacy, Sexism, and Patriarchy, we need to dismantle the current CS education model. We cannot just settle for one-off ethics courses that give students a flavor of how they should think about racialized biases in technologies [1]. Dr. Yadav will focus on the need to rebuild CS education away from explicitly focusing on technical competencies that prioritizes the tech industry needs rather than what's good for humanity.

REFERENCES

[1] Ruha Benjamin. 2019. Race After Technology: Abolitionist Tools for the New Jim Code. *Social Forces* 98, 4 (12 2019), 1–3. https://doi.org/10.1093/sf/soz162 arXiv:https://academic.oup.com/sf/article-pdf/98/4/1/33382045/soz162.pdf

[2] Carla Brodley, Benjamin J. Hescott, Jessica Biron, Ali Ressing, Melissa Peiken, Sarah Maravetz, and Alan Mislove. 2022. Broadening Participation in Computing via Ubiquitous Combined Majors (CS+X). In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1* (Providence, RI, USA) *(SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 544–550. https://doi.org/10.1145/3478431.3499352

[3] Mark Guzdial. 2013. Exploring Hypotheses About Media Computation. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (San Diego, San California, USA) *(ICER '13)*. ACM, New York, NY, USA, 19–26. https://doi.org/10.1145/2493394.2493397

[4] Aman Yadav, Marie Heath, and Anne Drew Hu. 2022. Toward justice in computer science through community, criticality, and citizenship. *Commun. ACM* 65, 5 (2022), 42–44.

[5] Aman Yadav and Marie K Heath. 2022. Breaking the Code: Confronting Racism in Computer Science through Community, Criticality, and Citizenship. *TechTrends* (2022), 1–9.

Using Computational Thinking to Demystify Computer Science for Elementary Teachers

Diane Levitt
Cornell Tech
Cornell University
New York, NY
diane.levitt@cornell.edu

Dylana Garfus -Knowles†
SOCS for All
Ashland School District
Ashland, OR USA
Dylana.Garfas@ashland.k12.or.us

Wyman Khuu
Science, Technology & Engineering
KIPP NYC
New York, NY
WKhuu@kippnyc.org

Sara Siddappa
PS 39 Cypress Hills CS
NYC Public Schools
New York, NY
sara.siddappa@chcs89.org

ABSTRACT

The computer science education research community has thought deeply about how students learn computational thinking (CT) as it relates to other domains of computer science (CS; e.g. programming) and core content areas (STEM, humanities), but less work has examined the role of CT in pathways to computer science for K-5 teachers. This panel examines the experiences of practitioners – educators, administrators, and curriculum designers--who have both experienced and supported others in incorporating CT in elementary school settings as a pathway to or component of computer science education. All panelists have worked with teachers not previously trained to teach CS and have encountered the many opportunities and difficulties of bringing CS to in-service teachers. They will reflect on the multiple ways educators grapple with CT: as an entry point to computer science, as a way to enrich core disciplines, and as a way to support equitable practice – for example, several of the panelists have experiences leveraging CT and other domains of CS to support the expression and development of emergent bilingual students. The panel will explore ways in which CT and its associated language and strategies for problem solving may provide a particularly helpful onramp to CS generally, including integration with other disciplines and with language about academic skills more generally.

ACM Reference format:

Diane Levitt, Dylana Garfus-Knowles, Wyman Khuu, & Sara Siddappa. 2024. Using Computational Thinking to Demystify Computer Science for Elementary Teachers. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3631658

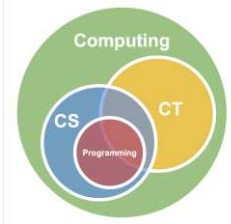
1 SUMMARY

Goals of the panel The panel will compare and contrast three efforts to using CT as a strategy to help normalize computing education:

- Help frame for the research community how teacher development in computing is approached at the elementary education level
- Provide multiple practitioner perspectives including curriculum developer, teacher, and the often overlooked role of principal
- Provide insight from multiple perspectives about the value and affordances of including CT in an elementary school education
- Share reflections from the field from three different geographical, demographic, and socioeconomic contexts in which CT and CS are being implemented
- Examine computing integration in multiple subjects, including English Language Arts, Math, Science, and for emerging bilingual students

Intended Audience K-12 CS education researchers, curriculum developers, educators, coaches, educational administrators, policymakers, funders

Relevance to the SIGCSE community This panel will present perspectives from the field about the role of CT in pathways to computer science for teachers, especially in settings focused on multilingual learners and other groups historically excluded from computing. We will examine the role of CT in making computing more accessible, and in creating a computing 'through line' via its integration into other disciplines for these students and their teachers. Panelists will present a discussion of how CT has made CS more accessible for elementary teachers.



The definitions of CT vary [1], though many refer to the definition first advanced by Cuny, Snyder, and Wing [2]. CT overlaps the discipline of CS, which includes "[t]he study of computers and algorithmic processes, including their principles, their hardware and software designs, their [implementation], and their impact on society" [3]. The diagram shown depicts CT as a subset of the general practices of computing and as a component of both CS and programming, but highlights how CT encompasses practices that lie outside of CS [4]. The CS education research community grapples with the pros and cons of CT in the elementary school setting. Some challenge

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03. https://doi.org/10.1145/3626253.3631658

the universal value of teaching CT [5]. However, CT, when offered as a component of computer science education, can create equitable access to computing literacies [6,7]. While students from more economically vulnerable communities may be exposed to coding, they are rarely offered the context of CT as part of computational problem solving [8].

2 PANEL STRUCTURE

Panelists will present their work independently, then begin an interactive discussion with the audience and one another.

- Panelists will introduce themselves and briefly describe their projects. They will each share their working definition for CT, the pedagogical underpinnings of its inclusion in their work, the affordances and challenges they faced, and any outcomes or findings.
- Panelists will be asked to comment on their perceptions of the role of CT in teacher preparation and student learning, of CS as well as core subjects.
- Panelists will be asked to comment on the role of CT in creating a sense of belonging for marginalized students, including students with disabilities.
- The moderator will check with the audience to surface any participants' experiences in implementing or researching the role of CT in K-12 classrooms and observations they can share, and pose any questions for the panelists.
- After this interactive discussion, panelists will pose questions for one another and audience members.

3 POSITIONS STATEMENTS

Moderator: Diane Levitt (Sr. Director of K-12 Education, Cornell Tech) is the PI for the NSF-funded grant CTCS: Integrating Computational Thinking into ELA and Math, Building an Onramp to Computer Science in Grades K-5

Position Statement: The cognitive and capacity overload of teachers, and their understandable resistance to a new discipline as well as a new set of teaching responsibilities, is a major barrier to the wholesale adoption of CS in elementary schools. Introducing computing in a progression that begins with CT, concepts that can tie to existing practice and their core values, and then follows with programming and impacts of computing, is proving to be a successful strategy for our public school partners.

Panelist 1: Dylana Garfas-Knowles, SOCS for All (Oregon) is an English Language Learning teacher in the Pacific Northwest. She is currently a lead teacher and coach for the NSF grant-funded project SOCS for All 2.0. In this project she is training teachers about CT and how to integrate CT concepts into their lessons and teaching strategies.

Position Statement: As a lead teacher/coach for integrating CT across subject areas in elementary education, I believe learning CT concepts is key to problem solving and creating a space for the foundation of CS. Providing access to this teaching/learning for all students (especially for Black, Latinx, Native American, Pacific Islander, and female peoples) is ultimately the way to open doors to CS for our students and provide more opportunity for diversity in the field of CS in the future.

Panelist 2: Wyman Khuu (Sr. Director of Science, Technology & Engineering, KIPP NYC) is an instructional leader and STEM curriculum designer for a large urban charter school network that includes 8 elementary schools and 9 middle schools serving over 7,000 students where they have been integrating CT into interdisciplinary classes across their schools.

Position Statement: At KIPP NYC, we believe that all students must create, innovate, and wonder every day in order to become future problem solvers. I believe that there is a great opportunity to leverage CT instruction in elementary school to expand who ultimately has access and agency in leveraging technology to solve the problems of today and the future. With intentional integration of CT into interdisciplinary subjects through authentic learning experiences, we can create the next generation of diverse learners and problem solvers.

Panelist 3: Sara Siddappa (PS 89, Brooklyn) is the assistant principal of a dual language immersion program serving Spanish-speaking migrant children in Brooklyn.

Position Statement: As the instructional leader in an elementary school serving an emerging bilingual population, we place a high priority on explicit language instruction for our students. Our school is focused on providing students with the skills and knowledge that will mitigate some of the racism and anti-immigrant bias they face outside their school, neighborhood, and homes. We have seen the power of student engagement in CT and CS, and the excitement our teachers feel when they bring this powerful discipline into their classrooms. We believe the language of computational thinking helps build a context for computing that goes beyond programming, and we are working to bring it into every discipline, in both English and Spanish.

REFERENCES

[1] Shuchi Grover and Roy Pea. 2018 Computational thinking: A competency whose time has come. In Computer science education: Perspectives on teaching and learning in school 19.1 19-38.

[2] Valerie J. Shute, Chen Sun, and Jodi Asbell-Clarke. 2017. Demystifying computational thinking. In Educational research review 22 142-158.

[3] Allen Tucker. 2003. A model curriculum for k--12 computer science: Final report of the ACM K--12 task force curriculum committee. ACM.

[4] Kelly Mills, Merijke Coenraad, Pati Ruiz, Quinn Burke, and Josh Weisgrau.2022.. Computational thinking for an inclusive world: a resource for educators to learn and lead. Digital Promise. https://digitalpromise.dspacedirect.org/bitstream/20.500.12265/138/5/CTForAnInclusiveWorld_DEC2021.pdf

[5] Peter J Denning. 2017. Remaining trouble spots with computational thinking. In Communications of the ACM 60.6, 33-39.

[6] Kathryn M. Rich, Aman Yadav, and Rachel A. Larimore. 2020. Teacher implementation profiles for integrating computational thinking into elementary mathematics and science instruction. In Education and Information Technologies 25, 3161-3188.

[7] Kimberley Gomez, Ung-Sang Lee, and Amy Berkhoudt Woodman. 2022. The role of teacher beliefs, goals, knowledge, and practices in co-designing computer science education curricula. In Teacher Learning in Changing Contexts. Routledge, 158-174.

[8] Jane Margolis, Joanna Goode, and Julie Flapan. 2017. A Critical Crossroads for Computer Science for All: "Identifying Talent" or "Building Talent," and What Difference Does It Make?. In Moving students of color from consumers to producers of technology. IGI Global, 1-23.

An Ethical Imperative: Increasing the Participation of Black Women in Computer Science Education Research

Tamara Pearson
Georgia Institute of Technology
Atlanta, GA, USA
tpearson30@gatech.edu

Stefanie Marshall
Michigan State University
East Lansing, MI, USA
slmarsh@msu.edu

Yolanda A. Rankin
Florida State University
Tallahassee, FL, USA
yrankin@fsu.edu

Mia S. Shaw
New York University
New York, NY, USA
mia.shaw@nyu.edu

ABSTRACT

Enhanced participation of Black women in CS education research is of ethical imperative, and empowering individuals who would otherwise not be able to fully engage in this community improves the quality of research and increases our national potential to solve real-world problems. We posit that the lack of participation of Black women in CS education research is not due to a lack of interest or expertise, but instead is another example of the continued marginalization of Black women that exists throughout our society. This panel of Black women scholars will share (1) their unique experiences and perspectives as Black women in the field of CS education, (2) the barriers Black women encounter in CS education research, and (3) existing and potential support mechanisms that enable Black women to more fully engage in the CS education research community.

ACM Reference format:

Tamara Pearson, Stefanie Marshall, Yolanda A. Rankin, & Mia S. Shaw. 2024. An Ethical Imperative: Increasing the Participation of Black Women in Computer Science Education Research. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3631660

1 SUMMARY

In order for the United States to remain competitive in the global market, it will need a diverse STEM workforce to tackle social, scientific, and technical problems that impact every aspect of our lives [3]. The need for a more computationally literate workforce led to the launch of the Obama Administration's Computer Science (CS) for All initiative in 2016. Unfortunately, despite a plethora of initiatives and a surge of research activity within the last six (6) years, racial and gender disparities in degree attainment within CS remain as the number of Black women persisting in CS continues to decline [2, 4, 5, 6]. These disparities also exist in the field of CS education research, where the majority of research is conducted by non-Black scholars.

Enhanced participation of Black women in CS education research is an ethical imperative. Empowering individuals who would otherwise not be able to fully engage in this community improves the quality of research and increases our national potential to solve real-world problems. Despite widespread diversification efforts, CS remains a lucrative, predominantly white and male field. If we consider that Black women educators, historically and contemporarily, frame their work through a Black feminist lens and center care, anti-oppression, and a responsibility for justice in their work, then the need for more Black women CS education researchers is clear [1]. We posit that the lack of participation of Black women in CS education research is not due to a lack of interest or expertise, but instead is an example of the continued marginalization of Black women that exists throughout our society.

2 PANEL STRUCTURE

This panel of Black women scholars will make the case for the need for more studies led by Black women CS education researchers. In particular, we will discuss the following questions:

1. What are the unique experiences and perspectives that Black women bring to CS education research?
2. What are the barriers that Black women encounter in the CS education research community?
3. What support mechanisms are effective in assisting Black women to engage more fully in the CS education research community?

The session will begin with 2-3 minute introductions by each panel member. This will be followed by a 45 minute conversation guided by the questions listed above. During this conversation we will highlight Care and Create, a summit designed to support Black women computer science education researchers. This will leave approximately 20 minutes for questions from the audience. Dr. Pearson will be the moderator as well as a participant on the panel.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3631660

3 POSITION STATEMENTS

3.1 Tamara Pearson

Tamara Pearson is the Constellations Center for Equity in Computing Senior Director of Research and Programs at the Georgia Institute of Technology. As a Black feminist scholar, Dr. Pearson recognizes that our identities shape the way we approach research and the topics we choose to explore. Her intersectional identity as a Black woman has shaped her continued interest in highlighting the experiences of Black women going all the way back to her graduate dissertation work. Her current work focuses on uncovering and disrupting the root causes for the inequitable access, participation, and success of historically marginalized communities in STEM and computing, with a specific focus on Black women. Dr. Pearson is the former inaugural Director of the Center of Excellence for Minority Women in STEM at Spelman College where she created and facilitated Care and Create, a retreat for Black women CS education researchers.

3.2 Stefanie Marshall

Stefanie Marshall is an Assistant Professor of Science Education at Michigan State University. Central to her scholarship, her justice work, is her intersectional identity as a Black woman and a "doer of science" – that is, she engages in inquiry and investigations, grounded in data, and guided by cultural knowledges and ways of being. Dr. Marshall originally drew on her policy, leadership, and science education background to examine CSEd policy questions. This led to her being first introduced to Black women in the CSEd community through a program called Care & Create, designed specifically for Black women in CSEd research. Collective time and space to be supported and nurtured encouraged Dr. Marshall to continue research in CSEd, specifically through a project called *Do CS Heroes Wear CAPEs? An Analysis of State Policy Infrastructures Designed to Support Equity-Focused Computer Science Education* funded by Google Computer Science-Education Research. Dr. Marshall's experiences exemplify the need for ways to encourage and embrace Black women scholars as they engage in research that disrupts the status quo.

3.3 Yolanda A. Rankin

Yolanda A. Rankin is associate professor in the School of Information at Florida State University. As a Black feminist scholar, Dr. Rankin leverages Black feminist epistemologies, or Black women's ways of knowing, as critical frameworks in computing education research. Her research develops and applies intersectional methods to advance the representation of Black women in the field of computing, contributing to the emerging subfield of intersectional computing. As an act of resistance to the marginalization of Black women in the field of computing, she invokes Black women's testimonial authority via interviews, storytelling, and sister circles to center their experiences navigating the computing ecosystem. Results from Dr. Rankin's research indicate how converging systems of power manifest and oppress Black women in the field of computing, more broadly, and the subfield of computing education research, specifically. Dr. Rankin was recently awarded the NSF Early CAREER award for her research entitled *Black Feminist Epistemologies: Building a Sisterhood in Computing*.

3.3 Mia S. Shaw

Mia S. Shaw is a Black woman artist, STEM educator, and learning scientist whose research interests include designing, implementing, and studying STEAM learning experiences that use interdisciplinary technologies and design that support (1) identity authorship; (2) speculative literacies; and (3) creative expression among Black and Brown youth. Throughout her scholarship, Mia has drawn from Black women's epistemologies and narrative practices as vehicles for interrogating technology's connections to systemic oppression and reimagining possible futures. Mia received her PhD from UPenn's Graduate School of Education in Teaching, Learning, and Teacher Education, and since fall of 2023, she has been an assistant professor/faculty fellow at NYU Steinhardt School of Culture, Education, and Human Development as an NYU Provost Postdoctoral Fellow.

4 ACKNOWLEDGEMENTS

We would like to acknowledge the Google Corporation for their funding of Care and Create, as well as all of the Black women scholars that participated in that event.

REFERENCES

[1] Tamara Beauboeuf-Lafontant. 2002. A womanist experience of caring: Understanding the pedagogy of exemplary Black women teachers. *The Urban Review* 34, 1 (2002), 71-86. DOI:https://doi.org/10.1023/a:1014497228517

[2] Xianglei Chen. 2013. *STEM Attrition: College Students' Paths Into and Out of STEM Fields* (NCES 2014-001). National Center for Education Statistics, Institute of Education Sciences, U.S. Department of Education. https://nces.ed.gov/pubs2014/2014001rev.pdf

[3] National Science Foundation. 2017. Women, minorities, and persons with disabilities in science and engineering. https://www.nsf.gov/statistics/2017/nsf17310/static/downloads/nsf17310-digest.pdf

[4] Stuart Zweben and Betsy Bizot. 2019. 2018 CRA Taulbee Survey. Computing Research Newsletter (CRN) 31, 5 (2019), 3–74.

[5] Stuart Zweben and Betsy Bizot. 2020. 2019 CRA Taulbee Survey. Computing Research Newsletter (CRN) 32, 5 (2020), 3–74.

[6] Stuart Zweben and Betsy Bizot. 2021. 2021 CRA Taulbee Survey. Computing Research Newsletter (CRN) Vol. 33, No. 5, May 2021

A Panel for Distinctive Computing Curricula: Programs Aligned with Liberal Arts Institutional and Departmental Identity

Andrea Tartaro
andrea.tartaro@furman.edu
Furman University
Greenville, SC, USA

Amanda Holland-Minkley
ahollandminkley@washjeff.edu
Washington & Jefferson College
Washington, PA, USA

Mario Nakazawa
nakazawam@berea.edu
Berea College
Berea, KY, USA

David Reed
DaveReed@creighton.edu
Creighton University
Omaha, NE, USA

ABSTRACT

Previous research argues that liberal arts computer science programs are defining new directions for curricula based on distinctive features of their institutional and/or departmental identity. These innovations, and the approach to curricular design that inform them, can inspire curricular innovation at large universities, community colleges, and K-12 institutions. This panel illustrates an approach to curriculum design based on institutional and/or departmental identity with specific examples from four programs. It will guide audience members to consider distinctive features of their program context and how that can influence their curricula.

ACM Reference Format:
Andrea Tartaro, Amanda Holland-Minkley, Mario Nakazawa, and David Reed. 2024. A Panel for Distinctive Computing Curricula: Programs Aligned with Liberal Arts Institutional and Departmental Identity. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3631653

1 SUMMARY

Liberal arts colleges are defining new directions for computer science (CS) curricula that can inspire curricular innovation at large universities, community colleges, and K-12 institutions [3]. The SIGCSE Committee on Computing Education in Liberal Arts Colleges has collected data showing that liberal arts institutions frequently make curricular design choices based on mission and distinctive features of the college's or department's identity [2]. The Committee is developing guidance to help institutions use this approach to review and redesign their curricula from this top-down perspective[1] while also considering curricular guidelines such as the forthcoming CS2023: ACM/IEEE/AAAI Computer Science Curricula [1]. This panel complements that work by illustrating how four programs have designed curricula guided by their program's distinctive features and providing interactive opportunities for the audience to consider curricula design based on their own program's unique context and values. Our goals for the panel include: (1) illustrating how programs have integrated features of institutional mission and departmental characteristics into their curricula; (2) guiding audience members in diverse institutional contexts to identify distinctive features of their context that could inform curricula; and (3) engaging in interactive "Q&A" with the audience that will spark new ideas about how curricula can reflect distinctive mission, identity, and values.

The intended audience for this panel is anyone interested in curriculum development. Faculty who are part of departments in liberal arts contexts will share some common ground with panel participants, while curriculum designers at all institution types can benefit from developing curricula that are more responsive to their local context. Faculty who are more familiar with curriculum design on a course-level will be encouraged to reflect on how a top-down approach to curriculum design offers opportunities to reconsider the content and approaches of individual courses. This panel is relevant to the entire SIGCSE community because computing education is still evolving to respond to current events, integrate emerging research, and consider best pedagogical practices. The panel is timely because the forthcoming release of CS2023 will lead many programs to review their curricula.

2 PANEL STRUCTURE

During 75-minute panel, the moderator will spend 15 minutes establishing context for the panel based on the previous work [2, 3] and using her own institution as an example. Each of the three panelists will discuss distinctive curricula at their institutions for 10 minutes. In the remaining time (30 minutes), we will use interactive "Q&A" to guide audience members to think about what makes the context of their program distinctive and how they can reflect that in their curricula.

3 POSITION STATEMENTS

3.1 Andrea Tartaro, Moderator, Furman University, Greenville, SC

Furman University's signature program, "The Furman Advantage," offers students "a personalized four-year pathway to graduation"

[1] https://computing-in-the-liberal-arts.github.io/CS2023/

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3631653

including "high-impact, engaged learning through research, internships and study away"[2]. To this institutional identity, the CS department aims to make connections across campus and support all students in our programs and course offerings. We offer both a traditional CS major as well as an interdisciplinary Information Technology major that allows students to apply classes in a "cognate discipline" . We are active participants in the University's Freshman Writing Seminar and two-year Pathways advising programs. We offer a thematic, breadth-based introductory course that is both the gateway course to our majors and fulfills Furman's liberal arts curriculum requirement for "Mathematics and Formal Reasoning". We offer courses fulfilling additional curriculum requirements, including "Historical Analysis," "Human Behavior," and "Ultimate Questions" . We have a thriving research and internship program including our "Computing in Community" program that connects students and faculty in research collaborations across campus. We offered a study abroad program on the history of technology and discovery . This year, we are implementing a new "Purposeful Pathways" requirement in our upper-level major courses that has students participate in co-curricular activities related to career exploration, professional preparation, networking & employer engagement, and engaged learning.

3.2 Amanda Holland-Minkley, Washington and Jefferson College, Washington, PA, USA

The Computing and Information Studies program was founded at Washington and Jefferson with the objective of creating a liberal arts computing program that would stand in contrast to a traditional CS major. This led to the creation of an interdisciplinary computing program that prepares students to use computing in support of any career or graduate education path they pursue and supports interdisciplinary uses of technology across the College. Our vision has a direct impact on the content and shape of our curriculum. Most importantly for our mission, we offer ten different no-prerequisite entry points into the CIS program. These courses all satisfy major requirements but also serve as useful standalone courses for students in other majors wanting to develop their computational skills. Major requirements are organized around skills such as "Analysis and Modeling" or "Design and Interaction" rather than content to allow students to customize their program. For faculty, this requires creative course design at the 100-level where freshmen through seniors are in class together and re-conceptualizing the course hierarchy as being based on a scaffolded pedagogy of project- and problem-based learning rather than a progression through content.

3.3 Mario Nakazawa, Berea College, Berea, KY

The CS Department at Berea College strives to create a welcoming environment with the firm belief that anyone can be a computer scientist. We work to lower barriers to the discipline, particularly for underrepresented minorities, and our introductory courses assume no prior computer programming experience. We employ variants of the Process Oriented and Guided Inquiry Learning (POGIL) as well as Service-Learning pedagogy, emphasizing the importance of team-based work with others of different backgrounds, opinions, and strengths. The Computer and Information Science (CIS) major has four concentrations. One allows for maximum flexibility for majors who want to explore connections with other disciplines, and the second that requires more depth in CS. The third and fourth concentrations deepen connections with mathematics and business, respectively. The CS faculty are constantly working with other departments for collaboration as computer technology becomes more pervasive. As a work college, all students work for at least 10 hours for Berea College, and some CIS majors have CS-related labor positions, such as Teaching Associates for CS classes, technicians in the IT department, or as developers who build and maintain various software platforms the College uses. Many more majors participate in internships and research opportunities during the summer to prepare for success upon graduation.

[2]https://www.furman.edu/furman-advantage/

3.4 David Reed, Creighton University, Omaha, NE

Creighton University is a comprehensive institution that includes a liberal arts college grounded in the Jesuit-Catholic tradition. The College of Arts & Sciences highly values *cura personalis* (care of the whole person) and encourages students to engage in the world through service and learning outside of the classroom. As a result, the size of each major is moderated to encourage minors, second majors, study abroad, and undergraduate research. With only 12 CS major courses, creativity is required to cover essential topics. For example, the major does not offer a separate discrete math or theory courses, but instead integrates topics in other courses (e.g., finite automata and graphs in Data Structures, computability and proofs in Algorithms). Since career paths in computing can vary considerably, the CS major has no external prerequisites. Instead, we rely on advising based on individual student goals and aspirations. In partnership with faculty, students can negotiate internships or independent studies/research which count as electives.

The Computer Science program is housed in a federated department that includes programs in Journalism and Graphic Design & Film. Majors in all three programs are required to take an intro computing course from CS faculty, a writing course from Journalism faculty, and a Web-based design course from Design faculty. Some elective courses (e.g., Social Media, Data Visualization) are cross-listed and can count across majors.

ACKNOWLEDGMENTS

The authors would like to thank Jim Teresco and the SIGCSE Committee on Computing Education in Liberal Arts Colleges.

REFERENCES

[1] ACM/IEEE-CS Joint Task Force on Computing Curricula. 2021–23. CS2023: ACM/IEEE-CS/AAAI Computer Science Curricula. https://csed.acm.org/

[2] Amanda Holland-Minkley, Jakob Barnard, Valerie Barr, Grant Braught, Janet Davis, David Reed, Karl Schmitt, Andrea Tartaro, and James D. Teresco. 2023. Computer Science Curriculum Guidelines: A New Liberal Arts Perspective. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 617–623. https://doi.org/10.1145/3545945.3569793

[3] James D. Teresco, Andrea Tartaro, Amanda Holland-Minkley, Grant Braught, Jakob Barnard, and Douglas Baldwin. 2022. CS Curricular Innovations with a Liberal Arts Philosophy. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1* (Providence, RI, USA) *(SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 537–543. https://doi.org/10.1145/3478431.3499329

Equitable Professional Learning for K12 Computer Science Teachers

Shana V. White
Dir, CS Equity
Kapor Foundation
Oakland, CA, USA
shana.white@kaporcenter.org

Allison Scott
CEO
Kapor Foundation
Oakland, CA, USA
allison@kaporcenter.org

Aman Yadav
College of Education
Michigan State University
East Lansing, MI, USA
ayadav@msu.edu

Diane Levitt
Dir of K12 Education
Cornell Tech
New York, NY, USA
diane.levitt@cornell.edu

ABSTRACT

To address racial and gender inequality in K12 STEM and computer science education, there needs to be mutli-pronged approach. In addition to the needed work for curricula that is culturally responsive and sustaining, inclusive of all student group, and positively welcomes the identities of historically marginalized people groups, K12 computer science teachers need to be equipped with the training and tools to implement the curricula, pedagogy, and instruction to mitigate the racial and gender gaps in K12 computer science education.

To address racial and gender inequality in K12 STEM and computer science education, there needs to be a multi-pronged approach. In addition to the needed work for curricula that is culturally responsive and sustaining, inclusive of all student groups, and positively welcome the identities of historically marginalized people groups, K12 computer science teachers need to be equipped with the training and tools to implement the curriculum, pedagogy, and instruction to mitigate the racial and gender gaps in K12 computer science education. Professional learning is an oft-used medium by curriculum providers as a means for teachers to adequately use boxed computer science curricula on computer science topics and content only. The panel proposes and will share how to implement culturally responsive-sustaining pedagogy within K12 teacher professional learning, the importance of the integration of culturally relevant computational thinking and computer science strategies for K12 teachers, improved understanding of instructional practices that benefit all students in the CS classroom, and equity centered instructional coaching can all greatly improve chances of decreasing racial and gender equity gaps at the K12 level for historically marginalized student groups. Much of the presentation will share current tools and programming available to K12 teachers as well as propose scalable and replicable models for use in in-service teacher education broadly, and in-service computer science and STEM teacher professional learning across the nation.

Keywords:
K12 computing education; culturally-responsive computing; equity; K12 prefessional development, K12 professional learning, K12 pedagogy and instructional strategies

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03. https://doi.org/10.1145/3626253.3631663

ACM Reference format:

Shana V. White, Allison Scott, Aman Yadav, and Diane Levitt. 2024. Equitable Professional Learning for K12 Computer Science Teachers. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3631663

1 SUMMARY

While calls for CS education are often framed in terms of national and economic competitiveness, there is a growing interest amongst equity scholars in curricular interventions that directly engage the ethical and sociopolitical issues surrounding CS and its role in society [1]. As a computer science education community, we must understand that the work of racial and gender equity in K12 computer science classrooms requires a multi-pronged approach. Not only must we improve overall access and opportunities for historically marginalized student groups to engage in quality computing education at the K12 level, but we must also ensure that our in-service teaching force is better equipped to meet the needs of diverse learners and able to engage in pedagogy that is inclusive and empowering of all learners.

In order to mitigate and eliminate the racial and gender equity gaps as well as increase accessibility for students with disabilities, our current K12 computer science teachers must be provided with more robust and extensive professional learning opportunities to grow and improve upon their professional praxis. This shift in K12 computer science professional development will require several steps including a clearer understanding by all K12 CS stakeholders of what is culturally responsive-sustaining computer science [2] and how to implement it more broadly within K12 classrooms and the transformation of K12 computer science instructional practices to include themes of relevance, confidence, identity, and social justice [3]. K12 computer science pedagogy and instruction provided by teachers should not only equip students with the knowledge and skills to innovate and effectively utilize tech but also expand students' critical thinking abilities, computing identities, and ethical reasoning skills to reflect upon their own biases and critique systems that exacerbate inequities in computing [4]. Educators of K12 computer science [5], better equipped with quality culturally responsive-sustaining pedagogy and equity center instruction will play a key role in revolutionizing approaches to computing education, ensuring equitable computer science classrooms, and significantly mitigating the racial and gender inequities that still currently exist in K12 computer science classrooms across the country. The data has shown that over a third of the current K12 computer science teachers are not equipped with the ability to engage in culturally

responsive-sustaining pedagogy within their classrooms nor able to tackle social justice and equity-based issues in CS with their students [6]. This panel addresses the question of "Where are we now in regards to K12 computer science professional learning, and where do we need to move toward?" as it relates to pedagogy, instructional practice, and research needs in building more criticality, culturally responsive-sustaining, equity centered, and student inclusivity within all of our K12 computer science education professional development for our teachers in order to truly make quality K12 computer science.

PANEL STRUCTURE

The panelists will start the session with a 15-minute overview of the current state of affairs in K12 computer science education nationally, as well as data about K12 teachers' perspectives from the Kapor Center, AIICE, and CSTA 2022 Teacher Report. Then, panelists will be given 25 minutes to present the work and scholarship they are doing related to culturally responsive computing professional learning, computer science instructional coaching at the K12 level, and additional tools and supports being developed to support K12 computer science teacher pedagogy and instructional practices. The session will conclude with an audience discussion, with anticipated questions ranging from the definition of culturally responsive-sustaining computer science to best practices in engaging in discussions around K12 professional learning models, and also to what research related to professional development needs in K12 computer science remains to be explored.

SHANA V. WHITE

Shana V. White is the Director of CS Equity Initiatives at the Kapor Center. Shana is a passionate educator who works as an advocate for marginalized groups in education and has an unwavering commitment to providing opportunities for historically marginalized students to engage in unique and relevant learning experiences. She believes in purposeful disruption of the status quo and encourages teachers to exercise autonomy and humanize schools for students. At the Kapor Center, she leads, advocates for, and supports local and national level work around culturally responsive-sustaining computer science pedagogy, teacher professional development, K12 CS curricula, racial justice work in K12 computer science education, and equitable K12 computer science school/district partnerships.

ALLISON SCOTT

Allison Scott is the CEO of the Kapor Foundation, part of the Kapor Center family of organizations, aiming to increase racial diversity in tech and entrepreneurship. At the Foundation, Allison leads efforts to: (a) conduct research on barriers and solutions to racial inequality in tech, and (b) operates programs and invest in pathways into the tech/entrepreneurship workforce, and (c) work in partnership with stakeholders to advocate for transformational change in policies and practices to expand racial equity in technology. She is currently a Principal Investigator on multiple national grants and has co-authored reports to expand equity in computer science education and increase participation of Black, Latine, and Native women across the computing pipeline.

AMAN YADAV

Aman Yadav is the associate director of Computing Education at Michigan State University as well as a Lappan-Phillips Professor of Computing Education. His research focuses on preparing teachers to integrate computational thinking to support their disciplinary instruction and also supporting high school CS teachers to bring equitable pedagogical approaches in their CS classrooms. He is currently leading a National Science Foundation project to train high school CS teachers to use culturally responsive-sustaining approaches in their CS Principles course. In addition, Aman has focused on bringing critical perspectives into CS education to help students understand how technologies can disproportionately harm individuals from marginalized communities.

DIANE LEVITT

Diane Levitt is the Director of K12 Education at Cornell Tech. She drives Cornell Tech's work with the NYC CS4All initiative with the NYC Department of Education as well as additional initiatives with the NYC computer science education community as well as nationally. She is responsible for Cornell Tech's annual conference, To Code and Beyond, convening educators, nonprofit and industry leaders, policymakers, and funders to examine best practices and innovative solutions in K-12 computing education.
Before joining Cornell Tech, Diane was the Director of Corporate Social Responsibility for Belkin, and philanthropic advisor to Belkin's founder and CEO. Diane received a Master's degree in Early Childhood Education from Bank Street College, and a Bachelor of Science in Women's Studies and Anthropology from Barnard College, Columbia University

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation, Kapor Center, Microsoft TEALS, Cornell Tech, Michigan State University CREATE for STEM, and Google.

REFERENCES

[1] Vakil, Sepehr and McKinney de Royston, M. 2019. Exploring Politicized Trust in a Racially Diverse Computer Science Classroom. Race Ethnicity and Education 22, 4 (2019), 545–567

[2] Kapor Center. 2021. CULTURALLY RESPONSIVE-SUSTAINING CS EDUCATION: A FRAMEWORK. Retrieved from https://www.kaporcenter.org/wp-content/uploads/2021/06/1_CRCSFramework-Report_v7_for-web-redesign-.pdf

[3] T. C. Madkins et al., "Culturally Relevant Computer Science Pedagogy: From Theory to Practice," 2019 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT), 2019, pp. 1-4, doi: 10.1109/RESPECT46404.2019.8985773.

[4] Vakil, S. & Ayers, R. 2019. The racial politics of STEM education in the USA: interrogations and explorations, Race Ethnicity and Education, 22:4, 449-458, DOI: 10.1080/13613324.2019.1592831

[5] Scott, K. A., Clark, K., Hayes, E., Mruczek, C., & Sheridan, K. 2010. Culturally relevant computing programs: Two examples to inform teacher professional development. In D. Gibson & B. Dodge (Eds.), Proceedings of Society for Information Technology & Teacher Education International Conference 2010 (pp. 1269-1277). Chesapeake, VA: AACE.

[6] Kapor Center, AIICE, CSTA. 2022. "Moving Towards a Vision of Equitable Computer Science: Results of a Landscape Survey of PK12 CS Teachers". Retrieved from https://csteachers.org/documents/en-us/1d16f7a7-e39b-4ebd-a5fa-b8cf82e6f029/1/

Using High School Student Perspectives to Develop an IoT-based CS Curriculum

Pramod Abichandani*
pva23@njit.edu
New Jersey Institute of Technology
Newark, NJ, USA

Craig Iaboni
New Jersey Institute of Technology
Newark, NJ, USA

Prateek Shekhar
New Jersey Institute of Technology
Newark, NJ, USA

ABSTRACT

This paper reports on the design, motivations, and preliminary development outcomes of an ongoing NSF-funded study to create an Internet of Things (IoT) based CS curriculum tailored for high school students. The curriculum adopts the 4-layer IoT model and emphasizes an immersive, hands-on pedagogy using single-board devices and the Python programming language. Prior to designing the curriculum, focus groups were conducted using Keller's ARCS model of motivation as a conceptual framework. Qualitative data analysis about student attention, relevance, confidence, and satisfaction revealed actionable insights used to design the curriculum. Preliminary curriculum implementation has underscored IoT's promise in engaging high school CS students and developing modern computing skills applicable to college and career pathways.

ACM Reference Format:
Pramod Abichandani, Craig Iaboni, and Prateek Shekhar. 2024. Using High School Student Perspectives to Develop an IoT-based CS Curriculum. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635610

1 INTRODUCTION AND BACKGROUND

Emerging technologies like the Internet of Things (IoT) present new opportunities for motivating high school student interest in computer science (CS) education.

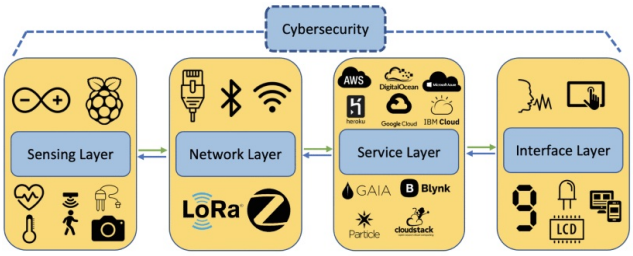
1.1 IoT-bsaed CS Education

The IoT platform, characterized by its integration of hardware, software, real-time data, and user interfaces, offers a practical approach for CS education [2]. Its relevance is heightened for grades 9-12 due to the focus on physical computing and the use of accessible programming languages such as Python and NodeMCU/LUA. These languages simplify the on-boarding process for grades 9-12 CS students. IoT-driven courses can leverage a mix of devices and tools to turn complex programming concepts into hands-on experiences. Readers are referred to [2] for a detailed literature review of IoT in CS/STEM education. In general, the IoT architecture can be consolidated into a 4-layer model as depicted in Figure 1 [3]:

(1) Real-world Sensing: Comprising sensors and actuators for data acquisition tasks and automation.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635610

Figure 1: The IoT 4-layer architecture was used as the guiding framework for curricular design [2].



(2) Communication Networks: Facilitating data relay and device inter connectivity.
(3) Software Services: Encompassing data processing, analytics, and inferential algorithms.
(4) User Interfaces: Interfaces for data visualization and system interaction.

Cybersecurity applies to all layers of IoT technologies. This model was used as a curriculum design framework encapsulating both IoT technicalities and related societal considerations [2].

1.2 ARCS Model of Motivation

Keller's ARCS model is a methodological framework grounded in the expectency-value theory, targeting four core motivational dimensions: attention, relevance, confidence, and satisfaction [4]. Specifically, these dimensions encompass:

(1) Attention: Capturing and maintaining student interest.
(2) Relevance: Aligning with individual objectives.
(3) Confidence: Enhancing student belief in their capability.
(4) Satisfaction: Offering internal or external rewards post accomplishment.

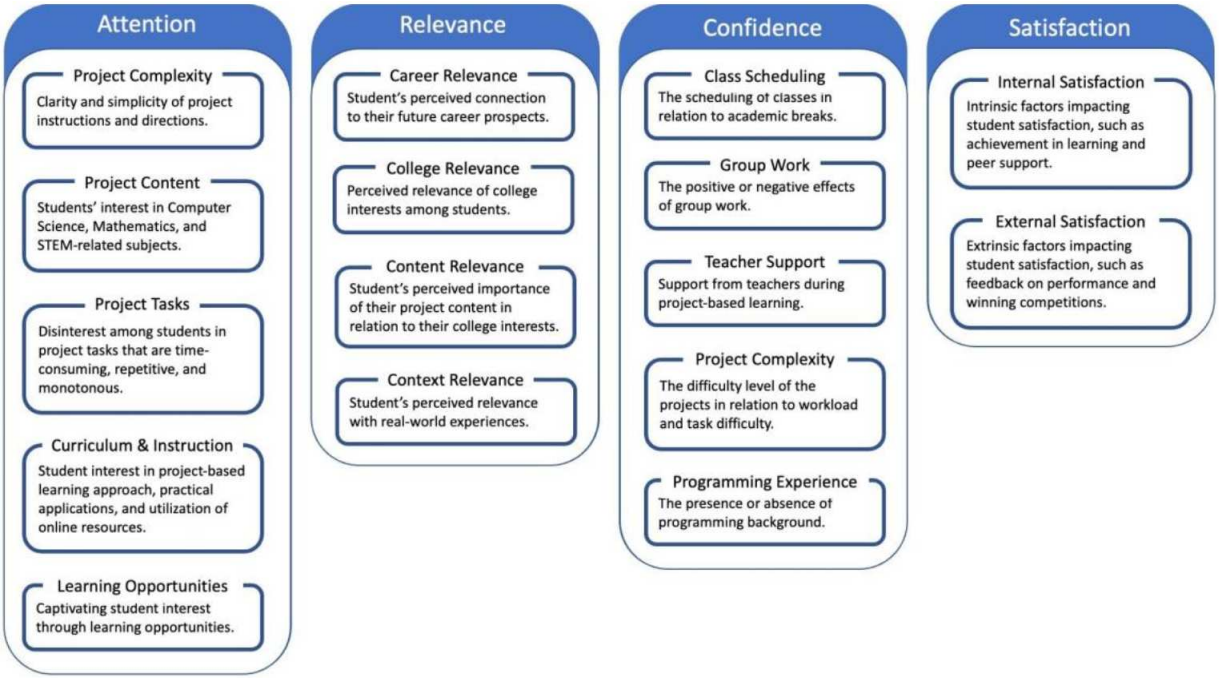
Several studies have employed the ARCS model to evaluate student motivation and the efficacy of pedagogical interventions [5–7]. The model underscores the imperative of addressing motivation, especially for promoting student engagement in learner-centric instructional methods.

2 ARCS FOCUS GROUP FINDINGS

The emergent ARCS factors and subfactors from the qualitative analysis of the focus group data are noted below. The summary of findings is depicted in Figure 2.

Attention. Students favored hands-on, group-based learning with explicit instructions. Interest heightened when making real-world connections through engineering instead of only CS topics. "It would be nice, if it like covered more, like more aspects of engineering, because a lot of what we do is just like coding stuff."

Figure 2: The summary of focus group findings.



Relevance. Students favored projects that held career significance in software development, meaning in preparing for college studies, and personal connection. Some students struggled to bridge coding concepts to tangible applications. "So personally, I think one of the most entertaining projects for at least an IoT kit was the joystick... And, you know, I like I like, you know, video games. And I think that's interesting to learn how they actually work."

Confidence. Educator guidance and peer support to build coding skills, and prior programming experience raised confidence, although new programmers felt challenged by variables, scope, and libraries. "I think what boosted my confidence in this class was just like working as a team, and working with other people, and being able to make something that you're proud of."

Satisfaction. Students felt satisfied mastering new programming paradigms, but errors sometimes dampened enthusiasm. "For me what's really satisfying is like once I grasp the concept and I know it like well, then I can assist my other classmates."

3 METHODOLOGY

Using the 4-Layer IoT model as a foundation, the curriculum incorporated the use of single-board computing devices like Arduino and Raspberry Pi, a sensor kit, and Python [1]. Two-week long curricular modules covered concepts and coding exercises related to all layers of the IoT model. Real-world connections were emphasized across projects based on focus group findings. Students presented Python code and demonstrated IoT systems weekly rather than having miditerm or final exams. The incorporation of student-led capstone projects wherein students picked their own project statement provided motivation. The curricular modules included:

(1) Getting started with Python and IoT sensor hardware
(2) Capturing and processing real-time sensor data
(3) Developing IoT interfaces and interactions
(4) Network protocols for IoT connectivity
(5) Cloud services for IoT
(6) Capstone projects: environmental and home automation

4 CONCLUSION

This ongoing study focused on integrating high school student perspectives to develop an IoT-centric curriculum for grades 9-12 CS education. Ongoing work involves grades 9-12 class deployment, iteration, and data collection on learning outcomes.

5 ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation (NSF) under Grant DRL- 2010259. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

REFERENCES

[1] P. Abichandani, D. Lobo, and P. Shekhar. An introductory internet of things curriculum for grades 9-12 computer science classes. In *2022 IEEE Frontiers in Education Conference (FIE)*, pages 1–6, 2022.
[2] P. Abichandani, V. Sivakumar, D. Lobo, C. Iaboni, and P. Shekhar. Internet-of-things curriculum, pedagogy, and assessment for stem education: A review of literature. *IEEE Access*, 10:38351–38369, 2022.
[3] L. Da Xu, W. He, and S. Li. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243, 2014.
[4] J. M. Keller. Development and use of the arcs model of instructional design. *Journal of instructional development*, 10(3):2–10, 1987.
[5] J. M. Keller. How to integrate learner motivation planning into lesson planning: The arcs model approach. *VII Semanario, Santiago, Cuba*, 1:1–13, 2000.
[6] K. Li and J. M. Keller. Use of the arcs model in education: A literature review. *Computers & Education*, 122:54–62, 2018.
[7] P. Shekhar, H. Dominguez, P. Abichandani, and C. Iaboni. Unpacking high school students' motivational influences in project-based learning. *IEEE Transactions on Education*, pages 1–11, 2023.

Evolution of an Integrated, Elementary CSforAll Curriculum

W. Richards Adrion
Manning College of
Information & Computer Sciences
University of Massachusetts
Amherst, MA, USA
adrion@umass.edu

Emrah Pektaş
College of Education
University of Massachusetts
Amherst, MA, USA
epektas@umass.edu

ABSTRACT

A research-practice partnership (RPP) used a teacher co-design process, supported by equity-focused professional development, to create an elementary-level curriculum that integrates content, practices, and learning progressions from state computing standards with other standards-based curricula. Most district students are part of historically marginalized groups and the RPP chose to develop an equity and inclusion-focused curriculum that would be taught in all elementary classrooms to all students. Twelve teacher teams, supported by researchers and ELL and SPED specialists, designed, piloted, and documented 23 modules of 4-8, 45-minute lessons across K-5. Early adopter teachers followed the pilots and implemented the modules in their classrooms with the goal of facilitating adoption by all elementary classroom teachers. After being interrupted by the pandemic, the RPP developed a strategy where principals in cohorts of schools agreed to collaborate with RPP school-based lead teachers to establish professional learning communities (PLCs) to support classroom implementation of the modules. Eleven schools participated in a 2021-22 cohort and nine more schools joined in 2022-23. Centering equity, PLCs, and quality module documentation and materials are key to sustaining and evolving the CSforAll curriculum. The modules were revised based on feedback obtained from ELL and SPED specialists, early adopters, teacher coordinators, researchers, and district curriculum directors. Using a large data set of meeting and classroom observation records, interviews, field notes, focus groups, surveys, and module documentation, we track the evolution of the curriculum and provide a detailed analysis of one module as an example.

ACM Reference Format:
W. Richards Adrion and Emrah Pektaş. 2024. Evolution of an Integrated, Elementary CSforAll Curriculum. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635495

1 SUMMARY

An NSF CSforAll RPP includes a major research university, a medium-sized, urban school district, and experienced evaluators. Across the district, 69% of 23,800 students are Hispanic/Latinx, 18% are Black or African American, and 85% come from low- income environments. Before the RPP began, students did not have access to computing curricula before high school where students of color seldom enrolled in available courses. To increase interest and reach all students, the RPP focused on the over 10,000 K-5 students in 32 elementary schools where all students can be engaged early in their education with an integrated, equity- centered elementary CS/CT curriculum taught by classroom teachers. The RPP avoided creating another "special" curriculum and instead created K-5 modules that integrate content, practices, and learning progressions derived from the computational thinking (CT) strand of the Massachusetts digital literacy and computer science standards [2] with other standards-based curricula in mathematics, English language arts, social studies, and science, technology, and engineering. The goal was to build equitable computer science and computational thinking expertise and confidence among classroom teachers so they could enable students to apply computational thinking in learning across subject areas.

As part of an iterative co-design process for the initial modules, five RPP teacher coordinators led the design process and later activities and continue to serve as a bridge between the researchers, teachers, and the district. The RPP recruited classroom teachers to co-design, develop, pilot, and assess modules. Eight-person teams were organized in dyads and supported by the teacher coordinators, ELL and SPED consultants, researchers, and the district instructional team. The dyads developed four modules per grade level in two grade levels each year using an iterative process where each module would be designed, piloted, assessed, and refined successively by three dyads. In each year following these pilots, the RPP recruited additional "early adopter" teachers who implemented the modules in their classrooms with the aim of facilitating adoption by all classroom teachers within the district. From 2018-19 through the 2020-2021 school year, the design teachers piloted 23 modules, 4 per grading period in kindergarten through grade 4 and 3 longer, project-based modules in grade 5. Table 1 summarizes the design process and indicates the data collected. While initial module design was largely influenced by PD and mentoring by researchers and coordinators, continuous curricular revisions were largely driven by classroom experience and feedback from the dyad partners, RPP coordinators, ELL and SPED consultants, researchers, and the district academic team.

The RPP continued its curriculum design, piloting, early adoption, and related professional learning over the first three

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635495

Period	Activity & Module documentation	Prof. Development	Revisions	Observations
2018-19	Four K & 3rd modules were designed, piloted, & documented during 3 iterations	Launch-CS PD without equity	During pilot iterations with input from SPED/ELL consultants who emphasized UDL and equity. During summer with input from design teachers and early adopters.	Pilot process and every iteration with field notes, videos & interviews. Researchers followed one module/grade level in 2018-19 observing all modules, one dyad/grade level in 2019-20, observing all but one module, and two grade 2 modules in 2020-21.
2019-20 Covid impact	Four 1st & 4th modules were designed, piloted, & documented during two in-school, one online iteration. Early adopters taught K & 3rd modules	Embedded PD covering CT, standards, devices, platforms, and equity for both design and early adopter teachers		
2020-21 Covid impact	Four 2nd & 5th modules (three in 5th) were designed, piloted, & documented during two-three online iterations. Early adopters taught K, 1, 3 & 4th modules online and after school			
2021-22	Modules taught in some classrooms in first cohort of 11 schools	Lead teacher PD covering modules, devices, and equity. PLCs with classroom teachers	During summer with input from classroom teachers	Artifact interviews in grade 4
2022-23	Modules taught in most classrooms in two cohorts of 11 + 9 = 20 schools			One classroom in 16 schools with field notes, videos & artifacts.
2023-24	Modules taught in most classrooms in three cohorts of 11 + 9 + 6 = 26 schools		Ongoing with input from teacher focus groups	10 planned and pending.

Table 1. A Summary of Curriculum, Design and Development, Revisions, Observations, and Supporting Professional Development

years through closure, online and hybrid learning, and in-class instruction with 145 design and early adopter teachers having taught the curriculum in all 32 elementary schools reaching approximately 2,000 K-5 students. The Covid impact, including pandemic disruptions, teacher shortages, and general stress on district personnel, prevented the planned full implementation of the curriculum, so the RPP proposed a "cohort" model to the district that would be implemented in three cohorts each year. Cohort school principals recruited RPP school lead teachers to form CSforAll professional learning communities (PLCs) in each school to provide support for the over 425 classroom teachers in 20 schools in the first two cohorts. The district hired computer science/technology coordinator (CS/TC) teachers and a district CS supervisor. Each principal, school lead, and CS/TC teacher collaborated to support classroom teachers in the cohort schools. The RPP supported short, twice yearly all cohort PD sessions and quarterly PD and 1-1 mentoring for the lead teachers.

Sustainability and ensuring equity depend on both the effectiveness of PLCs and the quality and useability [7] of the module documentation. Curriculum revisions were undertaken in collaboration with the RPP researchers and included explicit references to CS and the integrated discipline standards, improved content, sample scripts, common formatting, and equity and differentiation guidance in 2018-19, online adaptations in 2019-20, resolved links and copyright issues in 2020-21, simplified documentation and equity spotlights in 2021-22, and options for coordinating pacing and flexibility in 2022-23. Revisions were based on feedback and classroom observations. The full curriculum is posted on a district website and was reviewed and included in the Massachusetts education department DLCS curricula guide [1].

We specifically trace the evolution of a 2nd grade module as an example. This module was designed to allow students to understand how models may represent real-life solutions and how simulations can be used to test these solutions. Students researched natural disasters, gathered data, designed model houses that might withstand natural disasters, and compiled their experiences using a form of an engineering notebook implemented in Scratch. The authoring dyad had hoped to develop lessons, drawing on project equity PD, and applied the HILL [3] model by emphasizing student identities, developing skills, gaining knowledge, and understanding power, authority, and oppression. While the module promotes personal and engineering identities, the assumption that 2nd grade students would have direct experiences with significant natural disasters proved weak. Teachers were given more leeway in choosing scenarios in later revisions. This module was revised several times and lessons from this module were among those observed during 2023. These revisions did improve the useability of the module [7]. We saw a number of variations during classroom observations, however classroom teachers struggled with conveying CT concepts but were more comfortable the science content and promoting engineering identities. We analyzed the module from five perspectives: the state standards [2], two CT frameworks [4-5], a CT & science framework [6] and an equity framework [3]. We identified several shared computational thinking (CT) concepts and skills present in the module - abstraction and decomposition, modeling and simulation, debugging, data, and the creation of inclusive learning resources. Ongoing revisions are focused on providing more explicit guidance around CT concepts and equity.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grants DRL-1837086 & DRL-2219452. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Jake Foster, Melissa Zeitz, Lisa Manzi and David Petty. 2022. Massachusetts Digital Literacy and Computer Science Curriculum Guide. https://www.doe.mass.edu/stem/dlcs/curriculum-guide.pdf

[2] *Digital Literacy and Computer Science.* Massachusetts Curriculum Framework. 2016. https://www.doe.mass.edu/frameworks/dlcs.pdf

[3] Gholdy Muhammad and B. L. Love. 2020. *Cultivating genius: An equity framework for culturally and historically responsive literacy.* Scholastic.

[4] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proc. of the American Educational Research Association*, Vancouver, Canada, vol. 1, p. 25.

[5] Kelly Mills, Merijke Coenraad, Pati Ruiz, Quinn Burke, and Josh Weisgrau. 2021. *Computational thinking for an inclusive world: a resource for educators to learn and lead.* Digital Promise.

[6] Lautaro Cabrera, Diane Jass Ketelhut, Kelly Mills, Heather Killen, Merijke Coenraad, Virginia L. Byrne, and Jandelyn Dawn Plane. 2023. Designing a framework for teachers' integration of computational thinking into elementary science. *Journal of Research in Science Teaching.*

[7] Alaina Katherine Mabie, Monica McGill, and Brenda Huerta. 2023. A Systematic Literature Review Examining the Impacts of Integrating Computer Science in K-5 Settings." In *2023 Proc. of the ASEE Annual Conference & Exposition.* 2023.

Understanding the Role of Temperature in Diverse Question Generation by GPT-4

Arav Agarwal
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
arava@andrew.cmu.edu

Karthik Mittal
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
kkmittal@andrew.cmu.edu

Aidan Doyle
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
adoyle@andrew.cmu.edu

Pragnya Sridhar
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
pragnyas@andrew.cmu.edu

Zipiao Wan
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
zwan@andrew.cmu.edu

Jacob Arthur Doughty
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
jadought@andrew.cmu.edu

Jaromir Savelka
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
jsavelka@andrew.cmu.edu

Majd Sakr
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
msakr@institution.edu

ABSTRACT

We conduct a preliminary study of the effect of GPT's temperature parameter on the diversity of GPT4-generated questions. We find that using higher temperature values leads to significantly higher diversity, with different temperatures exposing different types of similarity between generated sets of questions. We also demonstrate that diverse question generation is especially difficult for questions targeting lower levels of Bloom's Taxonomy.

ACM Reference Format:
Arav Agarwal, Karthik Mittal, Aidan Doyle, Pragnya Sridhar, Zipiao Wan, Jacob Arthur Doughty, Jaromir Savelka, and Majd Sakr. 2024. Understanding the Role of Temperature in Diverse Question Generation by GPT-4. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635608

1 INTRODUCTION

Multiple Choice Questions (MCQs) are a common assessment tool in computing education. They allow for large-scale assessment, and can be graded automatically and, hence, efficiently.

However, developing effective MCQs remains an arduous task. Instructors need to devote significant time to craft question stems and answer choices in such a way that would allow for proper assessment of a student's understanding of the topic.

There has been significant recent work into using large language models (LLMs) such as GPT-4 to generate educational materials, from student code explanations to auto-grader feedback [5, 6]. One of the key parameters influencing the outcomes of the natural language generation of GPT models is `temperature`. The parameter generally determines the variety in generated text, with larger values generating more diverse text in other settings. Despite its key role, there have not been many studies dedicated to exploring its effects on educational content generation. In this paper, we provide early results from our investigation into the effects of different `temperature` settings in an MCQ generation pipeline. Existing literature suggests that content generated by GPT-4 can be homogenous; temperature provides a way to improve the diversity of the generated content with minimal prompt engineering [2].

2 METHODOLOGY

We developed a system that generates MCQs, utilizing OpenAI's GPT-4 [3]. Given a specific learning objective (LO), and high-level information about a course and a module, the system generates a single MCQ.

In order to evaluate the effects of different temperature settings on generated questions, we gathered a dataset of 52 LOs spanning across multiple levels of Bloom's Taxonomy and common topics in introductory CS topics. Bloom's Taxonomy provides a hierarchical categorization of educational goals, and provides useful context to MCQ generation systems [1]. For each LO, we generated 3 MCQs of a randomly-chosen question type (either Fill-In-The-Blank, Code Tracing, Recall, Scenario or Identify Correct Output) across several values of `temperature` (0.2,1.0,1.2). These values were chosen to contain both the default value for GPT-4 (1.0), and the largest and the smallest reasonable values suggested by our initial experiments. We found that larger temperature values than 1.2 lead to generally unusable questions, while smaller temperature values than 0.2 simply rephrased the same question most of the time. For each set of MCQs, we randomly sampled two instructors from a group of four and asked them to answer two questions: *Q1-distinct*: How many of the generated MCQs were distinct? and *Q2-complete*: Would it be possible to author an additional MCQ that is distinct from the generated ones but still aligned with the LO? *Q1-distinct* focused on assessing the diversity of the set of generated MCQs, while *Q2-complete* focused on assessing the completeness of the set.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635608

Temperature	1 Distinct Question	2 Distinct Questions	3 Distinct Questions
0.2	34	46	22
1	17	17	68
1.2	18	20	64

Temperature	N/A or I Don't Know	No	Yes
0.2	1	11	92
1	1	16	87
1.2	2	15	87

Table 1: Q1-distinct and Q2-complete vs. Temperature

Bloom's Taxonomy Level	Percentage of Question Sets with Any Duplicates
Create	32%
Analyze	54%
Apply	55%
Understand	53%
Remember	67%

Table 2: Percentage of Question Sets with Duplicates vs. Taxonomy Level

If an instructor is able to generate a distinct question, then there is potentially room for improvement, whereas if the instructor is unable to generate a new question then it is likely the LO is totally covered by the existing questions.

In total, we collected 313 annotations. Our overall inter-rater agreement was 0.30472 (Fleiss's κ), indicating we had fair agreement between instructors for both *Q1-distinct* and *Q2-complete* [4].

3 RESULTS

We present the results of the preliminary study in Table 1 and Table 2. The left half of Table 1 counts the instructor responses to *Q1-distinct* for each temperature setting, while the right half of Table 1 counts the instructor responses to *Q2-complete*. When temperature is set to 0.2, we tend to generate more sets of questions where either two of the questions were too similar or all three questions were too similar. When temperature is set at 1.0 and 1.2, we tend to generate more sets of questions where all three questions are distinct. Comparatively, the instructor responses to *Q2-complete* were the same at all temperature levels as, no matter the temperature setting, instructors generally answered that it was possible to create more distinct and aligned questions even with the generated question set.

We performed a chi-squared test and subsequent post-hoc analysis, and concluded that while the distributions of responses are significantly different comparing `temperature` 0.2 to the rest, there is not a statistically significant difference between setting the `temperature` to 1.0 and 1.2. This suggests that, for the purpose of picking `temperature` values for quiz generation, one should generally strive to work with larger `temperature` values.

Thus, in order to generate diverse questions using GPT-4, one should focus on temperature values between 1.0 and 1.2.

4 DISCUSSION

Besides the above quantitative evaluation, we also performed a qualitative evaluation to discover trends, focusing on MCQs targeting different cognitive levels of Bloom's Taxonomy.

We found that there appeared to be a relationship between the distinctness of MCQs and the targeted LO. If we look only at the level of Bloom's Taxonomy and look at the percentage of MCQ sets where *Q1-distinct* identified multiple distinct questions (Table 2), we can see that, as we go to higher levels of Bloom's Taxonomy, instructors generally view the questions as having fewer duplicates. However, there is no clear relationship between completeness of generation (*Q2-complete*) and Bloom's Taxonomy. This suggests the issue with generation is not a problem with the LO itself, but with the generation process, and that GPT-4 might not be as capable in generating diverse MCQs at lower levels of Bloom's Taxonomy compared to higher levels of Bloom's Taxonomy.

Thus, when looking at future studies in diverse question generation from LOs, one should focus on understanding LOs that are at lower levels of Bloom's Taxonomy, as they seem harder to generate automatically.

5 FUTURE WORK

Following this work, we would like to focus more heavily on various angles of the diverse question generation problem. Besides temperature, we would like to experiment with looking at the effect of other parameters like `frequency_penalty` for diverse question generation, alongside differing prompting and LLM chaining strategies.

REFERENCES

[1] Benjamin S Bloom, Max D Englehart, Edward J Furst, Walker H Hill, David R Krathwohl, et al. 1956. Taxonomy of educational objectives, handbook I: the cognitive domain. New York: David McKay Co.

[2] Paul Denny, Hassan Khosravi, Arto Hellas, Juho Leinonen, and Sami Sarsa. 2023. Can We Trust AI-Generated Educational Content? Comparative Analysis of Human and AI-Generated Learning Resources. arXiv:2306.10509 [cs.HC]

[3] Jacob Doughty, Zipiao Wan, Anishka Bompelli, Jubahed Qayum, Taozhi Wang, Juran Zhang, Yujia Zheng, Aidan Doyle, Pragnya Sridhar, Arav Agarwal, Christopher Bogart, Eric Keylor, Can Kultur, Jaromir Savelka, and Majd Sakr. 2024. A Comparative Study of AI-Generated (GPT-4) and Human-crafted MCQs in Programming Education. In *Proceedings of the 26th Australasian Computing Education Conference* (<conf-loc>, <city>Sydney</city>, <state>NSW</state>, <country>Australia</country>, </conf-loc>) *(ACE '24)*. Association for Computing Machinery, New York, NY, USA, 114–123. https://doi.org/10.1145/3636243.3636256

[4] J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (1977), 159–174. http://www.jstor.org/stable/2529310

[5] Stephen MacNeil, Andrew Tran, Arto Hellas, Joanne Kim, Sami Sarsa, Paul Denny, Seth Bernstein, and Juho Leinonen. 2023. Experiences from Using Code Explanations Generated by Large Language Models in a Web Software Development E-Book. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 931–937. https://doi.org/10.1145/3545945.3569785

[6] Pranjal Dilip Naringrekar, Ildar Akhmetov, and Eleni Stroulia. 2023. Generating CS1 Coding Questions Using OpenAI. In *Proceedings of the 25th Western Canadian Conference on Computing Education* (Vancouver, BC, Canada) *(WCCCE '23)*. Association for Computing Machinery, New York, NY, USA, Article 11, 2 pages. https://doi.org/10.1145/3593342.3593348

An Investigation on Task Difficulty: Does Task Difficulty Depend on the Technology Used in Task Completion?

Mahir Akgun
makgun@psu.edu
Pennsylvania State University
University Park, PA, USA

Sacip Toker
sacip.toker@atilim.edu.tr
Atilim University
Ankara, Turkiye

ABSTRACT

Previous research indicates that task difficulty (i.e., students' judgments on a task's complexity) impacts their task performance. However, whether students' perceived task difficulty changes depending on the technology they use when completing tasks is still under investigation. The present study aims to address this gap in the literature. One hundred twenty-three students completed the study procedures. Students were randomly assigned to one of four groups (one control group and three experimental groups). Students were not allowed to use any technology in the control group. In contrast, those in experimental groups were permitted to use one of the following tools: e-textbook, Google, and ChatGPT. Students in each group completed three tasks with different complexities in the same order. The data was analyzed using repeated-measures ANOVA. The study revealed a significant interaction effect between groups and task difficulty perceptions at three levels. In all groups, perceived difficulty increased as the task complexity increased, but the change in students' perceived task difficulty across three tasks was impacted by the tool used when completing the tasks.

ACM Reference Format:
Mahir Akgun and Sacip Toker. 2024. An Investigation on Task Difficulty: Does Task Difficulty Depend on the Technology Used in Task Completion?. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635602

1 FUTURE INFORMATION

The relationship between task complexity and learning has been documented in the literature. Compared to simple or easy tasks, complex tasks result in excessive demand on learners and produce higher cognitive load, making them difficult to deal with, but students learn more from complex tasks [4]. Given that students can perceive the same task differently, students' subjective judgment on the complexity of a task (i.e., task difficulty) has been investigated, and its impact on students' learning has been shown in the literature. For example, Nawaz et al. [2] showed that tasks perceived as easy or too difficult may lead to poorer learning outcomes, while moderate task difficulty may result in better learning outcomes. Scasserra [3] indicated that student performance would decrease as the perceived difficulty of the task increased. Overall, these studies suggest that perceived task difficulty can impact students' learning outcomes and that the difficulty of tasks can be optimized for students to achieve better learning outcomes.

While the influence of task difficulty on learning has been studied extensively, the role of technology in students' perceived task difficulty has been understudied. In this study, we aim to examine whether students' perceived task difficulty varies depending on the technology they use when completing tasks (e.g., ChatGPT, Google, and e-textbook).

2 METHOD

One hundred fifty-two students enrolled in the Information Systems department of a private university participated in the study. Twenty-nine students failed to complete all tasks in the study, so they were removed from the study. Of 123 students completing the study procedures, 40 were female, and 83 were male.

We created three conditions for the study using Bloom's Revised Taxonomy of Educational Objectives [1]. Students completed activities regarding 'privacy' in each condition in the following order: Condition 1 (C1), condition 2 (C2) and condition 3 (C3). In C1, students completed an activity that required lower-level cognitive processes (i.e., remembering and comprehending); in C2, students worked on an activity calling for medium-level cognitive processes (i.e., applying). Students were asked to propose a solution to the problem they identified in C3, which required them to use higher-level cognitive processes (i.e., assessing, evaluating, and creating). A sample task used in each condition is presented below.

(1) C1: Please answer the following question: What is the European Union (EU) General Data Protection Regulation (GDPR)? List the GDPR requirements that must be met by EU organizations when processing personal information.
(2) C2: Suppose that you work for a tech company. The company's management wants to make sure it has policies and procedures in place to protect the privacy of visitors to its website. You have been asked to provide advice to the company about what needs to be included in the policies to ensure that they comply with the EU GDPR.
(3) C3: Examine the Optimum Customer Privacy Notice and evaluate whether it complies with the EU GDPR. Base your evaluation on the GDPR requirements. After the evaluation, identify issues with the privacy policy (PR). Prepare a list of suggestions to improve the PR document.

Experimental pre- and post-tests with random assignment were employed. Students were assigned to one of four groups (one control group and three experimental groups) before they began to work on activities. Students were not allowed to use any technology in the control group. In contrast, those in experimental groups were

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635602

permitted to use one of the following tools: e-textbook, Google, and ChatGPT. Groups were formed based on their search experience, self-efficacy, privacy, and ChatGPT expertise. Four participant clusters were formed from these features in a two-step clustering approach. Each cluster's four input variable scores were sorted, and four of the closest scores were randomly assigned to the control or other three experimental groups (e-textbook, Google, and ChatGPT groups). Thus, all input variable scores were distributed uniformly between research groups. The random assignment also included gender distribution. Gender distribution was also taken into account during random assignment. The number of students assigned to each group is as follows: control group (N=31), e-textbook group (N=25), Google group (N=29), and ChatGPT group (N=38).

Upon the completion of the tasks in each condition, a questionnaire used to measure the task difficulty and the task interestingness was administered. The questionnaire contains six items: four items were related to the task interestingness, and two items were used to measure the task difficulty. A Likert-type scale from strongly disagree to strongly agree was used for all items. The total score from each component was used to measure the study variables. The higher the interestingness score is, the more positive the perception of the tasks the participant has. In contrast, the higher the difficulty score is, the more difficulty the participant perceives about the tasks. The overall research design is illustrated in Figure 1.

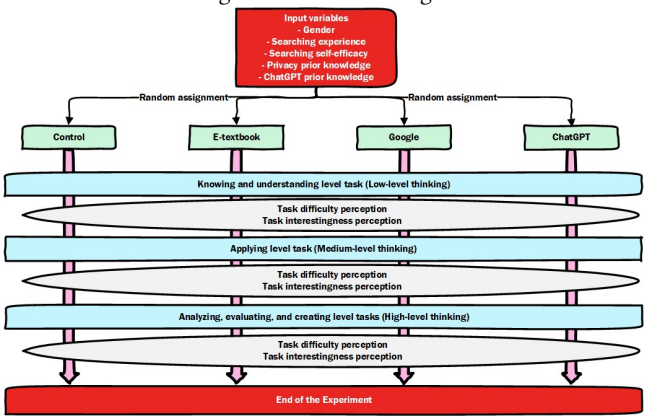


Figure 1: Research Design

3 DATA ANALYSIS

The data was analyzed using repeated-measures ANOVA with two separate measures. The first measure was the students' perceptions about the difficulty of each activity, and the second was their perceptions about the interestingness of the activities. Questionnaires were used to collect the data. Two items were used to assess difficulty, and four were employed to assess interestingness. The questionnaire was verified using explanatory factor analysis, and its reliability was ensured using internal consistency reliability, namely, Cronbach's alpha. The principal axis factoring generated two factors based on eigen values, and these factors explained 66.071% of the total variance. The first factor was task interestingness, and the item loadings ranged from .530 to .595. The second factor was task difficulty, and the item loadings ranged from .558 to .612. The Cronbach values were .675 and .606 respectively.

4 RESULTS

The study revealed a significant interaction effect between groups and task difficulty perceptions at three levels, $F(6, 238) = 2.419$, $p<.05$. On the other hand, the interaction between groups and task interestingness was not significant, $F(6, 238) =.547$, $p =.772$. In all groups, the difficulty perception increased as the task complexity increased. The increase observed in both Google and e-textbook is steep, while the ChatGPT group did not show a noticeable change in the difficulty perception during the first two tasks. However, there was a substantial change in the difficulty perception between the second and third tasks. Participants in the control group perceived a gradual rise in the task difficulty. The perceived difficulty was lowest in the ChatGPT group, followed by Google, e-textbook, and control groups (see Figure 2 for more details).

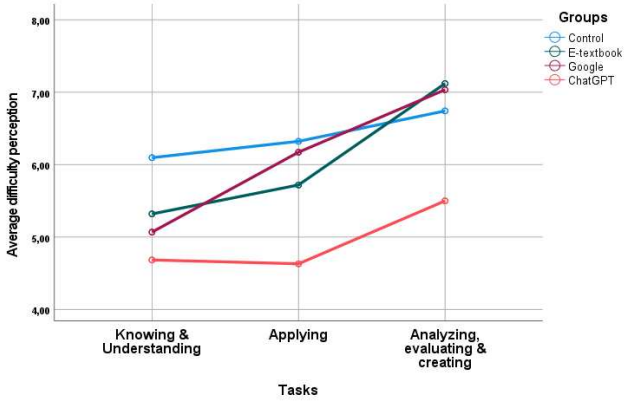


Figure 2: Changes in difficulty perception

5 CONTRIBUTIONS

This work contributes to the field by showing that students' perception of task difficulty is impacted by the technology used when completing tasks. Students in the ChatGPT group were the least affected by task difficulty. These preliminary findings suggest that using ChatGPT in computing education helps decrease students' perceived task difficulty, which in turn may increase their performance.

REFERENCES

[1] David R Krathwohl. "A revision of Bloom's taxonomy: An overview". In: *Theory into practice* 41.4 (2002), pp. 212–218.

[2] Sadia Nawaz et al. "Analysis of task difficulty sequences in a simulation-based POE environment". In: *Artificial Intelligence in Education: 21st International Conference, AIED 2020, Ifrane, Morocco, July 6–10, 2020, Proceedings, Part I 21*. Springer. 2020, pp. 423–436.

[3] Dominick Scasserra. "The influence of perceived task difficulty on task performance". In: (2008).

[4] Marie-Laurence Tremblay et al. "Simulation-based education for novices: complex learning tasks promote reflective practice". In: *Medical Education* 53.4 (2019), pp. 380–389.

Elementary Teachers Engaging with Learning Trajectories to Create Professional Learning Goals around Computer Science Integration

Jennifer Albert†
Zucker Family School of Education
The Citadel
Charleston, SC USA
jalbert@citadel.edu

Candace Joswick
Dept. of Curriculum
and Instruction
University of Texas at Arlington
Arlington, TX USA
candace.joswick@uta.edu

Deepti Joshi
Dept. of Cyber and Computer
Sciences
The Citadel
Charleston, SC USA
djoshi@citadel.edu

Robin Jocius
Dept. of Curriculum
and Instruction
University of Texas at Arlington
Arlington, TX USA
robin.jocius@uta.edu

Melanie Blanton
Zucker Family School of Education
The Citadel
Charleston, SC USA
mblanto1@citadel.edu

Robert Petrulis
EPRE Consulting
Columbia, SC USA
bpetrulis@gmail.com

ABSTRACT

In this poster, we present our efforts to engage elementary teachers with learning trajectories as a tool for developing both their own and their students' comprehension of computational thinking (CT) and strategies for integrating CT learning in their classroom. Eleven teachers, who voluntarily joined a teacher professional development (PD) program to develop teacher leaders for CT integration in the elementary context, attended a one-day PD session aimed at reviewing their knowledge of CT, participating in CT-infused lessons, and engaging with CT learning trajectories. Over the next year, teachers will participate in monthly virtual PD to continue to grow both their CT content knowledge and pedagogical knowledge. Our goal is to develop these teachers as teacher leaders who will support others as they integrate CT. This poster will show our current progress on CT learning trajectories and teacher leaders' responses to the tool.

CCS CONCEPTS

• Social and professional topics -> Computing education; Computing education programs; Computer science education

KEYWORDS

Teacher professional development, computational thinking, learning trajectories, elementary computer science education

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635579

ACM Reference format:
Jennifer Albert, Candace Joswick, Deepti Joshi, Robin Jocius, Melanie Blanton, and Robert Petrulis. 2024. Elementary Teachers Engaging with Learning Trajectories to Create Professional Learning Goals around Computer Science Integration. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635579

1 INTRODUCTION

The paper "STEM Education for the Future: A Visioning Report" [1], claims that "technology holds promise for creating equitable learning environments, but it also alters the skills we need in the future, and changes what and how we teach" (p. 24). Computational literacy, particularly in K-6 education, helps students build the necessary problem-solving and critical thinking skills to solve future problems [2]. To support teachers' professional learning around CT, they must have opportunities to practice the cognitive processes and activities in which they will engage students, but at a level tailored to their capacities as adult learners [3]. Additionally, teachers need opportunities to both investigate models of the instructional practices they are learning, and to enact those practices in increasingly complex environments [4]. This poster describes part of an ongoing teacher PD project for elementary teacher leaders.

2 BACKGROUND

In this section, we provide a brief background around teacher PD and teacher learning trajectories (LTs).

2.1 Teacher Professional Development

Research suggests that educational environments that center on students' funds of knowledge can improve equity in STEM disciplines [5]. To build culturally responsive, CS-infused curriculum, educators must understand their students' funds of knowledge and use them to position students as co-constructors of learning experiences that value home and community knowledge [6]. Aguirre et al. [7] suggest that teachers need multiple supports to use funds of knowledge as a regular part of curriculum design, including creating lessons that emphasize students' experiences, and STEM practices in the community setting.

2.2 Learning Trajectories

Learning trajectories (LTs) are ways of thinking and understanding a concept over time through interaction with specialized curricula and activities. As part of our MakingCT project (NSF #1923503), we developed a set of CT LTs for teacher CT integration development. They describe ways teachers think about CT integration in their classroom, including how to respond to students' thinking. The use of LTs in teaching and learning, and specifically in teacher development, has shown positive learning outcomes [8]. This poster will elaborate the CT LTs used during the teacher leader professional development and how the teachers plan to leverage them in their teaching as part of our UnboxingCT project (NSF #2300322).

3 METHODOLOGY

To understand teachers progress as they integrate CT into their disciplinary teaching, especially using LTs that describe how instruction may support students' knowledge of CT over time, we introduced teachers to CT LTs. Data from their self-reported perceived understandings of CT and conceptual thinking analysis of video-recorded PD sessions was used to inform teacher leader's initial impressions and potential uses of the CT LTs as both ways to direct their own growth and to support other teachers' development towards CT classroom integration goals.

Table 1. Selected Levels of Teacher LT for CT

Level	Level Description/Goal	Sample Teacher Practices for Observing Development
2	Able to teach existing lesson integrating CT	-Defines CT as specific concepts and/or practices -Teaches existing lesson with no or limited modifications (e.g., chunking lesson for time, classroom management)
3	Able to modify and co-create lessons	-Defines CT as problem-solving practices -Modification of existing lessons based on student CT learning needs

Video and audio recordings were transcribed and analyzed using qualitative coding techniques to investigate how teachers engaged with the LTs and their perceptions of the LTs to potentially set professional learning goals and reflect on practice. To preserve inter-rater reliability, two team members performed line by line process coding.

4 POSTER CONTENT

4.1 Evidence for learning trajectories

The teachers were asked to sort the "level descriptions" for the current LTs together in groups of 3-4 participants. Next, we shared how we currently organize them, and discussed similarities and differences. Initial analysis shows that many teacher leader groups did not organize their LT "levels" in the same way that the research team did. Many cited "Blooms Taxonomy" levels and "Depths of Knowledge" (DOK) as their method for organizing the levels and justified their thoughts by referencing that the looked at the "verbs". Then teachers discussed LTs in the context of CT-integrated lessons and how they may teach them in their classrooms. Teacher reflections also demonstrate how teachers used teacher LTs to understand their own learning: "The learning trajectory information was new and useful to think about. I have been thinking of coding paths for my students. The trajectory framework will help me to focus on what I want the students to learn and what's next."

4.2 Conclusions

Our initial work suggests that LTs can effectively support the design of integrated teacher learning experiences, as well as the practice of helping teachers categorize and examine their own learning about CT. Future research will unpack how teacher learning develops over time, and the ways that student LTs can help teachers understand student learning.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1923503 and 2300322.

REFERENCES

[1] M. Honey, et al. 2020. "STEM education for the future: A visioning report." *Tech. Rep.* 36.
[2] W. Martinez and D. LaLonde. 2020. Data science for everyone starts in kindergarten: Strategies and initiatives from the American Statistical Association.
[3] Hilda Borko 2004. Professional development and teacher learning: Mapping the terrain. *Educational researcher*, 33(8), 3-15.
[4] M Lampert, et al. 2013. Keeping it complex: Using rehearsals to support novice teacher learning of ambitious teaching. *Journal of Teacher Education*, 64(3), 226-243.
[5] Angela Calabrese Barton and Edna Tan. 2018. A longitudinal study of equity-oriented STEM-rich making among youth from historically marginalized communities. *American educational research journal* 55, 4, 761-800.
[6] Amy Wilson-Lopez, et al. "Latina/o adolescents' funds of knowledge related to engineering." *Journal of Engineering Education* 105.2 (2016): 278-311.
[7] Julia Aguirre, Maria del Rosario Zavala, and Tiffany Katanyoutanant. 2012. Developing robust forms of pre-service teachers' pedagogical content knowledge through culturally responsive mathematics teaching analysis. *Mathematics Teacher Education and Development* 14, 2, 113-136.
[8] Paola Sztajn, et al. 2012. Learning trajectory based instruction: Toward a theory of teaching. *Educational researcher* 41, 5, 147-156.

Achieving Diversity in AI-focused Graduate Research Traineeships

Cecilia O. Alm
Rochester Institute of Technology
Rochester, NY, USA
coagla@rit.edu

Esa Rantanen
Rochester Institute of Technology
Rochester, NY, USA
emrgsh@rit.edu

Kristen Shinohara
Rochester Institute of Technology
Rochester, NY, USA
kssics@rit.edu

Ferat Sahin
Rochester Institute of Technology
Rochester, NY, USA
feseee@rit.edu

Chelsea BaileyShea
Compass Evaluation
Rochester, NY, USA
chelsea@compassevaluation.com

Reynold Bailey
Rochester Institute of Technology
Rochester, NY, USA
rjbvcs@rit.edu

ABSTRACT

Our AI-focused traineeships for graduate students integrate research and education components to contribute to diversifying the AI research workforce. We describe the program and introduce multiple strategies to achieve interdisciplinarity, diversity, equity, inclusion, and accessibility. Early evaluation results are included.

ACM Reference Format:
Cecilia O. Alm, Esa Rantanen, Kristen Shinohara, Ferat Sahin, Chelsea BaileyShea, and Reynold Bailey. 2024. Achieving Diversity in AI-focused Graduate Research Traineeships. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635489

1 INTRODUCTION

Increasing the diversity of the AI research workforce presents a challenge in research education. We discuss strategies used in a recently introduced extracurricular traineeship program for graduate students in STEM that operates under an interdisciplinary, team science-focused [8], and apprenticeship-based [2] framework. The program aims to bridge multiple disciplines and promote interdisciplinary research. Open to students from all backgrounds, it focuses on pathways for STEM graduate students who are women, African American, Latino/a American, or Native American (AALANA), and deaf or hard of hearing, with research training and career-advancing professional development in AI. Early results suggest success in opening diverse pathways into AI.

2 SELECT RELEVANT PRIOR WORK

Diverse Disciplines in AI. Rapid expansion of AI technologies and applications has made it clear that the associated challenges are not merely computational, but also psychological, sociological, ethical, societal, and political. Future developers of AI technologies must therefore be educated in diverse disciplines, or be able to function in diverse teams with domain experts representing several disciplines [9]. Disciplinary spread is necessary for responsible AI development [3], and researchers with interdisciplinary backgrounds can better contribute to collaboration across fields [5].

Boosting Access for Students with Disabilities. Recent innovation in AI risks leaving behind people with disabilities. Two autoethnographic analyses of AI systems revealed gaps in systems that rendered them inaccessible [7, 15]. Despite efforts to combat barriers to participation in STEM [16], few students with disabilities, and in particular, students who are deaf or hard of hearing, pursue doctoral degrees in STEM and computing disciplines [6]. Lack of success in graduating students with disabilities can be attributed to technical and institutional inaccessibility [14].

Inequality in AI. It is also key to attract more AALANA and women researchers to AI; these groups continue to experience inequalities [1]. African American and Latino/a decision-makers are underrepresented in relevant fields [11], and similar underrepresentation impacts women [4]. Inclusion is critical for team success [13].

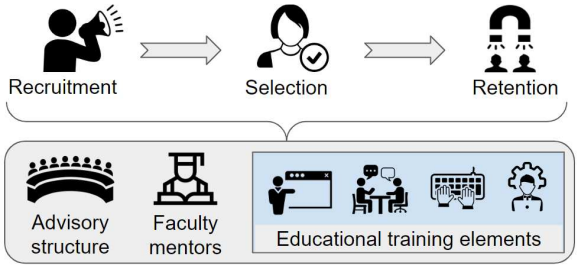


Figure 1: Procedural (top) and structural (bottom) diversification strategies.

3 STRATEGIES FOR DIVERSIFICATION

Figure 1 summarizes our strategies to achieve diversity, divided into *procedural* (top) and *structural* (bottom). In the program, students interact with mentors from multiple disciplines in four AI research areas and complete around a dozen educational training activities.

Recruitment, Selection, and Retention. To nurture diversity in the applicant pool, we use a multi-pronged approach with outreach to academic programs and virtual plus in-person open houses. The program has associated new graduate degree programs, enabling eligibility for students from many STEM programs, and it offers

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635489

incentives for underrepresented students. The selection process uses pre-set diversity targets in its final stages and has introduced a new diversification initiative, holding some slots for incoming PhD students from underrepresented groups, enabling PhD directors to claim these slots to attract the students into their programs.

We track one-year retention as a short-term measure of students who have not graduated and remain in good standing from one year to the next. This enables us to immediately detect any retention issue. Initial results indicate 100% retention of participants from all target groups. Retention strategies include (1) peer-mentoring and cohort-building events, (2) a student council representing all trainees at monthly meetings with the program team, (3) maintained trainee status with encouragement to continue with program activities after the focus year until graduation, and (4) the possibility to earn a certificate of completion.

Educational Training Elements. The program offers diversity and inclusion competency workshops with expert facilitators on *Building a Foundation for Inclusive Research* and *Diversity Statements*, and it ensures that all students who need access services are provided with them. In addition, within a retreat, the program has worked with a group offering a session with video material and a theatrical improvisation about diversity. Moreover, several students have participated in *Visits to Minority-serving Institutions* (MSIs) that approximate a campus interview visit experience. These short visits aim to provide opportunities for participants to: (1) present to external audiences, (2) promote research among undergraduates at the hosting institution in AI-focused research, (3) demonstrate graduate school opportunities, and (4) experience meeting with administrators at the host institution. Student visits to MSIs across the US have included a research presentation, one-on-one meetings with faculty and administrators, group meetings with labs, lunches with students, and meals with faculty. Trainee feedback has been positive, indicating that the visits meet program objectives.

Advisory Structure and Mentor Team. Internal and external boards, which advise program leadership, include members from the underrepresented groups that the program especially seeks to reach. In addition, the composition of the mentoring team intentionally emphasizes disciplinary and demographic diversity, positive mentoring [12], and belonging to the scientific community [10].

4 EVALUATION EVIDENCE AND DISCUSSION

Finally, we report on evaluation, considering participants' survey or focus group data. From the respondents (n=10; some questions unanswered), and using a 5-point scale from *not at all inclusive* to *extremely inclusive*, 78% felt the program was *extremely inclusive*. One participant commented: "The events held as part of [the program] were interdisciplinary, inter-cultural and very inclusive. They taught me more about integrating cultural aspects into the research domain." This aligns with program-external evaluation, which also summatively reported that "trainees were positive about the NRT's diversity and inclusion emphasis [...] and NRT-funded trainees spoke positively about their visits to MSIs and the value they derived from this experience." A paired t-test indicated a significant pre-to-post change with a large effect size for *inclusive career strategies* ($p = 0.009$), and for *recognizing cultural differences in a research setting* and *navigating cultural differences in a research setting* ($p = 0.02$, respectively).

Additionally, survey respondents reported that they *attended workshops on diversity and inclusion*, and 80% noted that "it sufficed," while 10% "wanted more," or felt "it was too much," and 90% confirmed they *participated in inclusive networking events*, with 60% noting that "it sufficed," while 30% "wanted more." Also, while 46% of respondents reported they felt *moderately, very or extremely prepared* for *writing a diversity statement for a job application* in the pre-program survey, this grew to 80% in the post-program survey. Trainee survey responses also showed pre-to-post knowledge growth for both *inclusive career strategies* and *promoting real inclusivity in AI research*, which grew from 62% or 46%, respectively, to 90% and *how to navigate working in diverse teams* from 92% to 100%.

Finally, 63% of faculty ($n = 8$) noted a *large gain* or *very large gain* that the program *increased [their] opportunities to mentor diverse students*. All responding faculty felt that students were *very prepared or extremely prepared* for *discussing diverse viewpoints*.

The evidence points to a positive impact so far. Continued data collection will allow us to improve the program as it expands.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Award No. DGE-2125362. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] S. N. Alegria. 2020. What do we mean by broadening participation? Race, inequality, and diversity in tech work. *Sociology Compass* 14, 6 (2020), e12793.
[2] A. Collins, J. Brown, and A. Holum. 1991. Cognitive apprenticeship: Making thinking visible. *American Educator* 15, 3 (1991), 6–11.
[3] V. Dignum. 2020. AI is multidisciplinary. *AI Matters* 5, 4 (2020), 18–21.
[4] P. Doerschuk, J. Liu, and J. Mann. 2009. Increasing Participation of Females and Underrepresented Minorities in Computing. *Computer* 42, 4 (2009), 110–3.
[5] S. Feng and A. Kirkley. 2020. Mixing patterns in interdisciplinary co-authorship networks at multiple scales. *Scientific Reports* 10, 1 (2020), 7731.
[6] National Science Foundation & National Center for Science and Engineering Statistics. 2017. *Doctorate Recipients from U.S. Universities: 2016.* (2017). https://www.nsf.gov/statistics/2018/nsf18304.
[7] K. S. Glazko, M. Yamagami, A. Desai, K. Avery Mack, V. Potluri, X. Xu, and J. Mankoff. 2023. An Autoethnographic Case Study of Generative Artificial Intelligence's Utility for Accessibility. *arXiv* (Aug. 2023).
[8] C. A. Hamilton, R. Vacca, and J. R. Stacciarini. 2017. The emergence of team science: Understanding the state of adoption research through social network analysis. *Adoption & Fostering* 41, 4 (2017), 369–390.
[9] D. Hannon et al. 2020. The education of the human factors engineer in the age of data science. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 64. SAGE Publications Sage CA: Los Angeles, CA, 480–484.
[10] B. L. Montgomery, J. E. Dodson, and S. M. Johnson. 2014. Guiding the Way: Mentoring Graduate Students and Junior Faculty for Sustainable Academic Careers. *SAGE Open* 4, 4 (2014), 2158244014558043.
[11] S. Rangarajan. 2018. Bay Area tech diversity: White men dominate Silicon Valley. *Reveal News* (2018).
[12] R. D. Robnett, P. A. Nelson, E. L. Zurbriggen, F. J. Crosby, and M. M. Chemers. 2018. Research mentoring and scientist identity: Insights from undergraduates and their mentors. *International Journal of STEM Education* 5 (2018), ar41. https://doi.org/10.1186/s40594-018-0139-y
[13] J. Shih. 2006. Circumventing discrimination: Gender and ethnic strategies in Silicon Valley. *Gender & Society* 20, 2 (2006), 177–206.
[14] K. Shinohara, M. McQuaid, and N. Jacobo. 2021. The Burden of Survival: How Doctoral Students in Computing Bridge the Chasm of Inaccessibility. In *Proceedings of CHI 2021* (Yokohama, Japan). Article 376, 13 pages.
[15] P. Smith and L. Smith. 2020. Artificial intelligence and disability: too much promise, yet too little substance? *AI and Ethics* 1, 1 (Oct. 2020), 81–86.
[16] L. P. Thurston, C. Shuman, B. J. Middendorf, and C. Johnson. 2017. Postsecondary STEM Education for Students with Disabilities: Lessons Learned from a Decade of NSF Funding. *J of Postsecondary Education and Disability* 30, 1 (2017), 49–60.

Designing Create Tech:

A Discussion of Previous Camp Iterations and Outcomes

Charlotte Avery
Department of Computer Science, University of Maryland
College Park, Maryland, USA
cjavery@umd.edu

ABSTRACT

Since 2017, the Iribe Initiative for Diversity and Inclusion in Computing (I4C) located at the University of Maryland conducted one to three high school programming one-day workshops each semester for students from local and surrounding communities. Trends in enrollment and participant feedback indicated that an extended learning experience over several consecutive days was a growing interest among participants from underrepresented populations in computing. These participants possessed little to no computing skills and were in the early stages of deciding whether to pursue the STEM field. To address these trends, in 2019, I4C launched two one-week summer camps, Introduction to Computing 101 and Make Your Own Wearables. Later, in 2022, Create Tech a two-week summer camp was launched to address concerns with previous camps' length, instructor preparation, and program design. The purpose of this poster is to discuss the previous camp iterations and the outcomes that led to the creation of Create Tech.

CCS CONCEPTS

• human-computer interaction • K-12 Instruction • programming

KEYWORDS: wearable technology, informal education, computer science education

ACM Reference format:

Charlotte Avery. 2023. Designing Create Tech: A Discussion of Previous Camp Iterations and Outcomes. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/1234567890

1. INTRODUCTION

Education systems across the U.S. are finding ways to offer computer science courses and integrate computational thinking across disciplines [1]. Currently, 53% of high schools offer a computer science course, up from 35% in 2018 [2]. However, many students in the post-pandemic era of K-12 education, especially those from historically underrepresented and disadvantaged groups in computing face challenges: (1) limited access to computing devices and the internet, (2) limited to no access to qualified teachers through teacher prep programs and professional development, (3) lack of interest and limited engagement with CS education, and (4) lack of consistent evidence-based curriculum and pedagogical support throughout their CS education journey [3]. These challenges quickly erode the pipeline to careers in STEM.

Informal programs, such as camps, provide students with experiential and authentic learning [4]. These opportunities allow students "to engage deeply with concepts and systems, to solve complex problems, to work collaboratively, and to be entrepreneurial and innovative [4]" which are 21st-century skills needed to pursue STEM careers.

Introduction to Computing 101

Introduction to Computing 101 (Intro) launched in 2019 and was designed as a one-week camp for rising 9th-12th graders who had little to no prior computing experience. Students would be introduced to Python programming which included variables, classes, data types, functions, loops, and conditionals. The goal of this program was to develop students' programming skills while growing their interest and confidence in pursuing the STEM field. From 2019 - 2021, there were a total of 46 students enrolled.

Make Your Own Wearables

Make Your Own Wearables (Wearables) launched in 2019 and was designed as a one-week camp for rising 8th-12th grade girls, who had previous experience with coding and computing topics, and expressed interest in the STEM field. Students would explore computing topics in wearable technology, and create their wearable technology using Circuit Playgrounds, Arduino, and sewing. The goal of this program was to strengthen students' programming skills through hands-on experience within a same-gender learning environment while growing their interest and confidence in pursuing the STEM field. In 2019, there were a total of 12 students enrolled.

Students in both camps learned about careers in computing, attended a field trip, and listened to various guest speakers who spoke about how to prepare to enter the profession. Due to the pandemic, Intro and Wearables were not offered in 2020. Since then, Wearables has been discontinued. In 2021, Intro returned as a virtual two-week summer camp.

3. DISCUSSION

When analyzing the participants' pre- and post-surveys data from the summer of 2019, we learned that students who strongly agreed with the statement "I am confident in my ability to develop programming skills" increased by 2% for Intro and there was no change for Wearables. We also found that students who strongly agreed with the statement "I can excel in a STEM field" increased by 11% for Intro and there was no change for

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.36356322

Wearables. Although participants in Intro reported having more confidence in their ability to develop programming skills and excel in a STEM field, the percentage from the start to the end of camp showed little significance. When participants and instructors were asked about their overall experience, a student shared, "This was a great camp and there were a lot of cool projects, but the end of the camp was a bit fast-paced." Another student shared, "The TAs didn't know what they were doing and were not helpful. Some of the instructors were also bad at explaining simple concepts." Instructors from both camps shared that the camp length negatively impacted their ability to teach the programming language effectively and to provide opportunities for students to apply programming language skills to real-world situations. Due to these findings, I4C's staff and instructional team extended both camps' length to two weeks. The audience remained students with little to no prior computing experience. Additionally, instructors' and Teaching Assistants' training included professional development in CS pedagogy, Lesson Planning, Classroom Management and Student Engagement in virtual spaces and discussions about Student Learner Profiles. Lesson Demonstrations and Observations were conducted to support instructors with effectively teaching computing topics.

The responses from summer 2021 pre- and post-surveys for previous statements "I am confident in my ability to develop programming skills" and "I can excel in a STEM field" increased by 11% and 6% respectively which showed some improvement since 2019. Furthermore, students' responses to "What was the most exciting concept/skill you learned at this camp?" showed that 84% of participants felt learning the components of Python was the most exciting. The data suggests that the camp's extended length and the changes to include teacher pedagogy to the instructor and TA training positively made an impact. However, when compared to all the students in Intro and Wearables camps in 2019, 42% mentioned applying computing skills was the most exciting. A student shared, 'It was really cool to use conductive thread to make plushies that could light up and make a sound" was their exciting skill. Another student shared that, "learning how to make games using code" was their exciting skill. One could infer that the camp length is extended, instructors are given more time to plan and create content where students learn program skills and are able to apply computing skills then there will be increases in all these areas for students with little to no prior computing skills.

4. CREATE TECH

In the summer of 2022, I4C launched Create Tech, a two-week non-residential summer program for rising 8th–12th graders from historically underrepresented groups in computing, who have some to no prior experience in coding and computing topics, and who may be in the early stages of deciding whether to pursue the STEM field. This program combined the best practices learned from both Intro and Wearables. Students create wearable and interactive technology using Python, Circuit Playground Express, raw materials (ie, Fabric, wood, plastic), and equipment (ie. 3D printer, sewing machine, woodcutter).

Program Design

Create Tech uniquely combines all the best practices from previous iterations into one camp. Create Tech has a low barrier to entry with no prior computing skills needed, university faculty members who specializes in human-computer interaction and immersive media design support content development, guest lecture, and share current research projects. Other faculty and staff support instructors and teacher assistants with training in pedagogy covering areas in Bias and Representation, Student Learner Profile, and Inclusive Teaching Strategies. Also, Lab Directors of Small Artifacts and Immersive Media Design support provide demonstrations and tours. Students attended field trips at Artech House and Planet Word to view exhibits of examples of immersive interactive technologies. Students have access to university resources such as the Singh Sandbox, where they select raw materials to create prototypes to later present in the program. All students finish the program with a circuit playground express and all the materials they used to create their project to continue learning.

Outcomes

Over the past two years, Create Tech's total enrollment was 24 in 2022 and 53 in 2023. There was a combined percent increase of 53% in confidence in their ability to develop programming skills and 21% in confidence to excel in a STEM field. Students shared, "The instructors and TAs were very helpful in giving them a raise" and "Circuit boards are the fundamental part of any piece of tech and the functions of it can be changed BY ME!"

5. FUTURE RESEARCH

A semi-structured focus group interview should be conducted after the completion of the third year of Create Tech. As students gain more access to computing opportunities informal programs will need to become more innovative in their design.

ACKNOWLEDGMENTS

This researcher would like to acknowledge the University of Maryland for the opportunity to access previous research and the ability to design and implement a new camp. Additional acknowledgments are given to my colleagues Dr. Jan Plane, Dr. Kristina Kramarczuk, and Kate Atchison who have supported all efforts in getting this paper submitted.

REFERENCES

[1] Yang, D., Baek, Y., Ching, Y.-H., Swanson, S., Chittoori, B., & Wang, S. (2021). Infusing Computational Thinking in an Integrated STEM Curriculum: User Reactions and Lessons Learned. European Journal of STEM Education, 6(1), 04. https://doi.org/10.20897/ejsteme/9560

[2] Code.org (2022), State of computer science education: Accelerating action through advocacy. Retrieved on October 13, 2023 from https://advocacy.code.org/2022_state_of_cs.pdf

[3] The Brookings Institute (2022). Building skills for life: How to expand and improve computer science education around the world. Retrieved on October 13, 2023 from https://www.brookings.edu/wp-content/uploads/2021/10/Building_skills_for_life.pdf

[4] Herbert, C. and Jenson, J. (May 2020). *Making in schools: Student learning through an e-textiles curriculum.* Discourse: Studies in the Cultural Politics of Education, (41(5), 740-761 https://doi.org/10.1080/01596306.2020.1769937

Considerations for Improving Comprehensive Undergraduate Computing Ethics Education

Grace Barkhuff
grace.barkhuff@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Jason Borenstein
borenstein@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Daniel Schiff
dschiff@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Judith Uchidiuno
jiou3@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Ellen Zegura
ewz@cc.gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

ABSTRACT

Computing Ethics (CE) courses are an increasingly important component of the undergraduate computing curriculum because of the outsized influence of computing on society. CE encompasses topics from multiple disciplines including the humanities; however, it is typically taught by educators within a Computer Science (CS) department in most undergraduate institutions in the United States, potentially leading to a less than comprehensive CE education for students. We surveyed 318 computing educators in US higher education to investigate CS educators' perceptions of how CE topics should ideally be taught. Most of our respondents thought that CE should be taught by a multidisciplinary team of instructors, and further that it should be taught both as a standalone course and embedded in other courses. Our research provides insights into ways to improve CE education that result in a better student experience.

ACM Reference Format:
Grace Barkhuff, Jason Borenstein, Daniel Schiff, Judith Uchidiuno, and Ellen Zegura. 2024. Considerations for Improving Comprehensive Undergraduate Computing Ethics Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635557

1 INTRODUCTION

Computing ethics is a vital part of an undergraduate's computing education. There are calls for more responsible computing education [5], and in order for an undergraduate computing program to receive accreditation, students must be able to "...make informed judgments in computing practices based on legal and ethical principles" [1]. CE may be included in the curriculum as a standalone course where the sole focus is on ethical content, as an embedded practice where ethical content is included in a non-ethics course, or as a combination of both teaching methods.

Several universities have explored a variety of approaches to teaching CE. For example, Harvard university introduced the Embedded EthiCS program, which uses the embedded method and has philosophy graduate students and postdoctoral fellows teaching ethics modules [4]; Stanford university experimented with a "deeply multidisciplinary approach" where they taught a standalone ethics course with instructors from philosophy, political science, and computer science [6]; and Brown University used a model where undergraduate teaching assistants provided support for embedding responsible computing into multiple courses [2].

An analysis of 202 technology ethics courses conducted by Fiesler et al. found that most technology ethics courses are taught within the computer science department [3], and a recent survey of 138 higher education computing instructors by Smith et al. found that 71.5% of respondents *disagreed* with the statement that CE should be taught by non-computing educators [7]. Despite these findings, CE involves non-computing subjects such as philosophy. Towards a more balanced undergraduate CE education, we explored the following research questions in our study: According to computing educators, (1) who should teach undergraduate CE, and (2) what teaching format is ideal for CE? We present findings from a survey of undergraduate computing educators from the United States (US), some with experience teaching ethics and others without.

2 METHODS

We conducted an online survey, administered through Qualtrics, of 318 undergraduate computing educators from the US. 82.4% (n=262) had experience teaching CE and 17.6% (n=56) did not. The survey comprised questions related to why participants did or did not teach CE or include CE in their courses and how they think CE should be included in the curriculum. The survey was distributed through a variety of e-mail lists available to the authors, project advisors, and their networks. Responses were collected between December 2022 and April 2023. Participants were compensated via a gift card raffle. The study was approved by the authors' Institutional Review Board.

3 RESULTS

Who should teach ethics? Figure 1 summarizes the responses received. Most respondents (67.2%, n=178) indicated that a team of instructors from multiple disciplines would be best suited to teach CE. This was the most common response regardless of whether

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635557

the respondent had experience teaching CE or not. 21.5% (n=57) responded that only computing faculty should teach CE. Of those who selected "Other," 31.6% (n=6) mentioned that what matters most is that those who teach ethics have firsthand experience with the subject, such as those who have encountered ethical issues at work. One respondent shared that the best person to teach is one with "firsthand experience good or bad... that's the only way I would pay attention so I assume it's the same for students. It's not real otherwise."

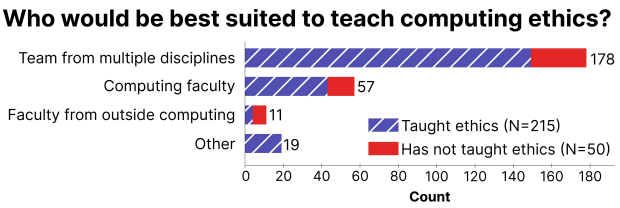


Figure 1: Participant responses to who should teach ethics

What teaching format is ideal? Figure 2 summarizes the responses received. A majority, 57.1% (n=153), think ethics should be taught both as a standalone course *and* as part of one or more technical courses. 25.4% (n=68) of respondents selected that CE should be taught using the embedded method. Of the 21 open-response comments received, 9 mentioned that ethics should be included in every course. One participant stated, "Every class should have some discussion of ethics because all courses and topics (even the most purely technical) influence people's lives."

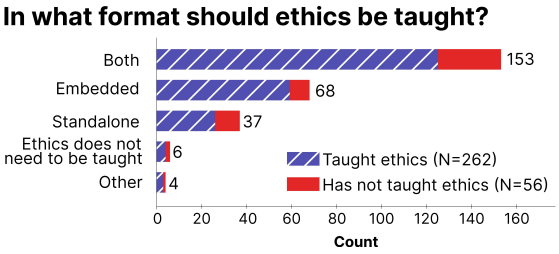


Figure 2: Participant responses to the ideal format for CE

4 DISCUSSION AND CONCLUSIONS

Our findings show that respondents believe CE should be taught by multidisciplinary teams and that CE should be integrated both as a standalone course and embedded into other technical courses.

The first finding provides nuance to that of Smith et al. which found that 71.5% of surveyed computing instructors did not believe CE should be taught by educators from outside computing [7]. Our data are in agreement: only 4.2% (n=11) of our respondents thought that faculty from outside computing are best suited to teach CE. However, when provided with the option for a team from multiple disciplines, that option is preferred. Although Smith et al. found that 67% of respondents to their survey felt confident in their ability to add ethics to their courses, some respondents felt their expertise in the area was limited and preferred to leave ethics teaching to others [7]. One possible explanation is that CE may not have been a topic covered in educators' own coursework. Therefore, these topics may not be ones that they have expertise in, leading to a desire for a team to teach CE in order to collectively be knowledgeable about both computing and ethics. However, as current students transition to faculty positions, this may no longer be the case.

Our respondents strongly favored teaching CE both as a standalone course and embedded in technical courses. This combined approach allows students to reap the benefits of both approaches, including multiple touch points throughout the curricula for breadth and a standalone course for depth. In the embedded approach, a CS educator is needed to teach the main CS content in an otherwise technical class. In order to use the team approach, educators from outside the CS department could guest lecture, or teaching assistants could teach the ethics content such as in Harvard's Embedded EthiCS approach [4]. An approach for standalone courses could be to team-teach with instructors from across disciplines. Alternatively, using a non-computing instructor for the standalone course and computing instructors for the embedded courses could achieve a similar team-based approach.

Teaching CE using a team and in both formats is not without challenges. Computing educators are already stretched thin, and additional CE takes time away from technical topics. However, the high amount of support for multidisciplinary and multiple format CE education justifies the need for cross-institutional CE teaching resources, additional educators, and increased institutional support to adequately prepare computing students to make sound professional judgments based on well-balanced ethics principles.

ACKNOWLEDGMENTS

This work is funded in part by National Science Foundation (NSF) Ethical and Responsible Research grant (SBE-2124745).

REFERENCES

[1] 2023. Criteria for Accrediting Computing Programs, 2023 – 2024. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2023-2024/.

[2] Lena Cohen, Heila Precel, Harold Triedman, and Kathi Fisler. 2021. A New Model for Weaving Responsible Computing Into Courses Across the CS Curriculum. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 858–864. https://doi.org/10.1145/3408877.3432456

[3] Casey Fiesler, Natalie Garrett, and Nathan Beard. 2020. What Do We Teach When We Teach Tech Ethics? A Syllabi Analysis. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 289–295. https://doi.org/10.1145/3328778.3366825

[4] Barbara J. Grosz, David Gray Grant, Kate Vredenburgh, Jeff Behrends, Lily Hu, Alison Simmons, and Jim Waldo. 2019. Embedded EthiCS: Integrating Ethics Across CS Education. *Commun. ACM* 562, 8 (2019), 54–61.

[5] Cathy O'Neil. 2017. Opinion | The Ivory Tower Can't Keep Ignoring Tech. *The New York Times* (Nov. 2017). https://www.nytimes.com/2017/11/14/opinion/academia-tech-algorithms.html

[6] Rob Reich, Mehran Sahami, Jeremy M. Weinstein, and Hilary Cohen. 2020. Teaching Computer Ethics: A Deeply Multidisciplinary Approach. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 296–302. https://doi.org/10.1145/3328778.3366951

[7] Jessie J. Smith, Blakeley H. Payne, Shamika Klassen, Dylan Thomas Doyle, and Casey Fiesler. 2023. Incorporating Ethics in Computing Courses: Barriers, Support, and Perspectives from Educators. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 367–373. https://doi.org/10.1145/3545945.3569855

Should We Teach Computing Ethics to Master's Students?

Grace Barkhuff
grace.barkhuff@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Judith Uchidiuno
jiou3@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Ellen Zegura
ewz@cc.gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

ABSTRACT

Computing ethics courses are generally considered essential in undergraduate programs, but there is limited research on the importance of their inclusion at the master's level. In this study, we extend research conducted by Dexter et al. in 2013 about the importance of ethics in master's programs. We surveyed Associate Chairs and Associate Deans of Graduate Studies representing 81 different computing programs in the United States. 72% of respondents shared that ethics should be required in master's computing programs. In contrast, only 46% of the 81 programs surveyed require any type of ethics course at the master's level. Our findings also uncover barriers to the inclusion of ethics at the master's level and propose strategies for alleviating these challenges.

ACM Reference Format:
Grace Barkhuff, Judith Uchidiuno, and Ellen Zegura. 2024. Should We Teach Computing Ethics to Master's Students?. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635553

1 INTRODUCTION

Computing ethics courses are part of the undergraduate curriculum at many institutions, owed in part by requirements for accreditation [2] and curricula recommendations from ACM and IEEE [5]. These courses discuss topics vital to the impact of computing on society, including topics such as privacy and surveillance, inequality, and environmental impact [4]. Courses covering ethics can be offered in multiple modalities, for example as a standalone course where the topic of the course is exclusively computing ethics or using the embedded ethics model where an existing course adds modules or assignment questions centered on ethics.

Research has been conducted to understand the best methods and modalities for teaching computing ethics, but most are focused at the undergraduate level. At the master's level, Trim and Nishad describe two course assignments that they used to teach ethics. They argue that the course should be required and that it serves as a "bridge" for students who are transitioning into computer science [7]. In addition, Dexter et al. explored faculty and student opinions on teaching ethics at the graduate level. They surveyed 55 faculty and 249 students from four universities in the United States (US) to investigate their approach to ethics education at the graduate level. None of the four universities surveyed required a standalone ethics course. Of the 55 faculty surveyed, 49% (n=27) favored a required ethics course, and 72% (n=40) favored an elective ethics course [3]. This poster updates and extends this study by surveying program administrators from a much broader set of institutions to understand their opinions on, barriers to, and their current approach to computing ethics education at the master's level.

We investigate the following research questions: In computing master's programs, (1) what is the current state of teaching computing ethics, (2) what is the stance of program administrators towards how/if computing ethics should be taught, and (3) what are the barriers to teaching computing ethics?

2 METHODS

We surveyed program administrators (Associate Chairs and Deans of Graduate Studies) in master's computing programs in the US asking if their program(s) offered ethics courses, and their opinions on teaching ethics at the master's level, including barriers to doing so. We received 81 valid responses, consisting of 81 unique programs from 67 unique universities. 70.4% (n=57) of programs represented were computer science programs. Responses were collected in June and July of 2023. The study was approved by the authors' Institutional Review Board. Participants were not compensated.

3 RESULTS

Current Ethics Offerings. We first asked participants about the ethics offerings for the programs they represent. Only 12.3% (n=10) of surveyed programs require a standalone ethics course. 37.0% (n=30) stated that their program requires a course with at least one ethics module (the embedded ethics model). There is some overlap between the two, as a total of 45.7% (n=37) stated that their program requires either a standalone ethics course or a course with an embedded ethics module or both. However, several respondents were unsure if their program had courses with ethics modules (19.8% - n=16). This uncertainty may be explained by changes made within curricula or courses by faculty who may add or remove an ethics module without program administration being aware, particularly when there is not an explicit focus on including ethics across the curriculum.

Attitudes Toward Teaching Ethics at the Master's Level. We asked participants if they thought ethics should be required at the master's level and whether it should be an optional elective. 72.2% (n=52) agreed that ethics should be required at the master's level, 80.6% (n=58) agreed that ethics should be offered as an optional elective course at the master's level, and only 4.2% (n=3) disagreed with both questions. Many respondents mentioned in free response comments that they believe ethics to be necessary. For example, one stated, "When designing and building software systems, ethics is essential

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635553

and complicated. Not helping students learn about ethical considerations in a Software Engineering curriculum is unimaginable." Some provided reasons why they felt ethics should not be required, such as lack of faculty expertise, not wanting to spend more time in the curriculum on non-technical topics, and not wanting to make any courses in the program required.

Barriers to Including Ethics. We surveyed participants about factors that prevented their programs from including or increasing the amount of ethics offered. The top five responses were: "Adding ethics would require increasing the course load or removing another requirement" (n=27), "It would take time to develop curricula for ethics" (n=24), "Ethics was already covered at the undergraduate level" (n=22), "Faculty don't have the right expertise to teach ethics" (n=21), "Ethics is already sufficiently covered in our master's program" (n=19). Figure 1 summarizes the responses collected. This data shows that although program administrators agree that including ethics in graduate programs is important, there are barriers to realizing this. One respondent summed this up by saying, "It is work to incorporate ethics, but we are working on it with our students."

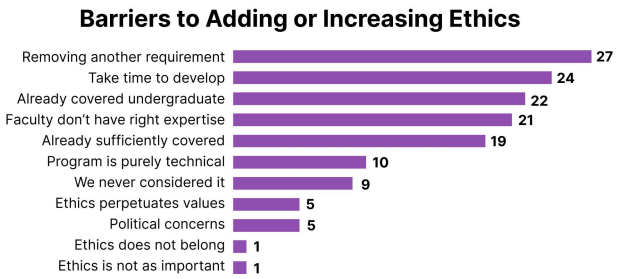


Figure 1: Participant responses to what prevents you from adding or increasing ethics in your program

4 DISCUSSION AND CONCLUSIONS

In 2010, Dexter et al. found that 49% (n=27) of surveyed faculty favored a required ethics course, and 72% (n=40) favored an optional ethics elective [3]. Our research finds a much higher percentage of participants in favor of ethics options, with 72.2% (n=52) in favor of required ethics and 80.6% (n=58) in favor of optional ethics. The reasons for this increase are likely multifaceted. For one, computing ethics has remained an important topic over the past ten years, and discussions of issues such as data privacy and the use of popular artificial intelligence tools have increased the urgency of computing ethics. Alternatively, we surveyed a distinct population from that surveyed by Dexter et al. Given that we surveyed Associate Chairs and Associate Deans of Graduate Studies rather than general faculty members, it is possible that individuals in this position favor adding ethics to the curricula more than general faculty members. Regardless, the high value placed on teaching computing ethics in our survey shows its importance for master's curricula today, and the low number of schools requiring standalone ethics at the master's level (12.3%) represents a disconnect between what program chairs want and what is currently feasible in their programs.

Although a larger number of programs (37.0%) in our survey offer embedded ethics at the master's level, this is insufficient. Embedded ethics has many merits, such as signaling the importance of the topic to students [6], however, we only asked if programs contained at least one course with at least one ethics module. Without continued, repeated, and sustained inclusion of ethics in multiple courses, students do not reap the benefits of the embedded ethics model.

Barriers to adding or increasing ethics content must be addressed to achieve a future where master's programs include ethics in their curricula. The top concern that adding ethics would necessitate removing another requirement can be alleviated through an embedded ethics program that is audited to ensure repeated ethics touch-points. The concern that ethics takes time to develop or that faculty do not have the right expertise can be addressed through the creation of shared repositories of ethics content. Faculty who are computing ethics experts can contribute course content and assessments for others who are non-experts to use. Some of this work is already underway and should be continued, such as the assignments published by the Internet Rules Lab [1]. Finally, the third most commonly cited barrier is that ethics is already covered at the undergraduate level. This is a false assumption, given that many students come from undergraduate institutions that do not cover ethics or from non-computing programs. Dexter et al.'s survey found that a substantial number of graduate students in their survey had not received ethics instruction at the undergraduate level [3]. This underscores the importance of including it at the master's level to ensure students are prepared for the ethical challenges they will encounter in their careers.

Overall, our survey uncovered strong support for including ethics at the master's level, but a limited number of schools have committed to and implemented this into the curriculum. By working together as a community dedicated to producing computing professionals educated with well-balanced ethics principles, we can overcome the barriers to doing so and better prepare students for their future in computing.

ACKNOWLEDGMENTS

Thank you to Rachel Bellamy, Anind K. Dey, and Mira L. Wolf-Bauwens for their insights on this project.

REFERENCES

[1] [n. d.]. Ethics-Based Computer Science Assignments. https://www.internetruleslab.com/ethicsbased-computer-science-assignments
[2] 2023. Criteria for Accrediting Computing Programs, 2023 – 2024. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2023-2024/.
[3] Scott Dexter, Elizabeth Buchanan, Kellen Dins, Kenneth R. Fleischmann, and Keith Miller. 2013. Characterizing the Need for Graduate Ethics Education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 153–158. https://doi.org/10.1145/2445196.2445245
[4] Casey Fiesler, Natalie Garrett, and Nathan Beard. 2020. What Do We Teach When We Teach Tech Ethics? A Syllabi Analysis. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 289–295. https://doi.org/10.1145/3328778.3366825
[5] CC2020 Task Force. 2020. *Computing Curricula 2020: Paradigms for Global Computing Education.* Association for Computing Machinery, New York, NY, USA.
[6] Barbara J. Grosz, David Gray Grant, Kate Vredenburgh, Jeff Behrends, Lily Hu, Alison Simmons, and Jim Waldo. 2019. Embedded EthiCS: Integrating Ethics Across CS Education. *Commun. ACM* 562, 8 (2019), 54–61.
[7] Michelle D. Trim and Hadi Nishad. 2019. We Learn by Doing: Modeling Inclusive Pedagogy in a Graduate CS Ethics Course. In *2019 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. 1–2. https://doi.org/10.1109/RESPECT46404.2019.8985698

Sokoban: an Assignment for an Object-Oriented and GUI Programming Course

João Paulo Barros
joao.barros@ipbeja.pt
Polytechnic Institute of Beja, School of Technology and Management, Beja, Portugal
Center of Technology and Systems (UNINOVA-CTS) and Associated Lab of Intelligent Systems (LASI), 2829-516 Caparica, Portugal

ABSTRACT

Motivating assignments is a well-known way to promote deeper learning in programming courses. As those assignments must also be aligned with the intended learning outcomes and preferably carry some novelty, the search for adequate assignments is permanent. This poster proposes a classical 2D one-person game that provides a solid basis for discussing and applying object-oriented concepts, GUI programming, and simple patterns. It was applied in an introductory object-oriented course. Students' feedback was highly positive, and anecdotal evidence suggests improved motivation and learning. Additionally, the game can be easily extended with some extra requirements and is amenable to different degrees of scaffolding. Hence, it can be used in more introductory or more advanced courses.

CCS CONCEPTS

• **Social and professional topics → CS1**.

KEYWORDS

education; assignment; homework; cs1; pedagogy; oop; programming; GUI; event-driven programming

ACM Reference Format:
João Paulo Barros. 2024. Sokoban: an Assignment for an Object-Oriented and GUI Programming Course . In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635493

1 INTRODUCTION

Programming courses strongly emphasize the practical application of problem-solving techniques through the completion of coding exercises and projects. These activities require students to utilize various concepts, frameworks, and patterns to achieve their objectives. Motivating tasks hold significant relevance as a means to foster experiential learning.

Games are a domain of application with the inherent advantage of being accessible to all students, as they possess the requisite baseline knowledge. Moreover, games are characterized by their ability to captivate and stimulate learners, offering a level of challenge that promotes engagement (e.g. [2]).

Additionally, GUI-based assignments are seen as "more like the real world, improve skills, more challenging" [1]. However, it is also imperative that they are in accordance with the desired educational objectives. Hence, the instructor faces a perpetual quest for engaging, aligned, and novel assignments. In this paper, we propose an assignment that was implemented and has yielded positive outcomes in a second programming course within a Computer Science curricula.

2 BACKGROUND AND OBJETIVE

Sokoban is a classical game available all over the internet in several forms (e.g. [6]). Hiroyuki Imabayashi created the game in 1981, and it was initially released in December of 1982 [4, 7] . It is a puzzle video game in which the player attempts to move boxes into storage areas by pushing them around in a warehouse. The game occurs inside a board (map) made of square positions. Each position (square) can only contain the player, a box, or an unpassable wall. Some free positions, the same number as the number of boxes, are marked as the final destination for a box. The player has to push the boxes in one of four directions (up, down, left, right) so that all are positioned in the marked locations. The difficulty comes from the boxes' locations relative to the final positions, the walls in between, and the number of boxes, as only one can be pushed in each move and only to a free position. Hence, it is also a map game where the map is the board with some walls and marked final locations, but where all the map is known from the beginning. The moving objects are the player and the boxes. The games can have several levels. To that end, it is enough to define additional maps. Luckily, there are numerous maps on the internet as the maps cannot be random if one wants to be sure there is a solution.

The assignment's general objective is to implement the classical motion planning interactive game. This objective is achieved by a set of listed requirements given to students. Those requirements are aligned with the intended learning outcomes. More specifically, students learn object-oriented programming and design, as well as GUI programming, and receive an introduction to one simple MVC architecture and file processing. It forces the use and creation of multiple classes and objects and can be used to promote information hiding, encapsulation, modularity, abstract types, inheritance, and polymorphism.

The difficulty can be adjusted by the provided base code, required features, or both.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635493

3 APPLICATION

The assignment was implemented during the academic year 2022/2023 as part of a course on introductory object-oriented programming. The course is available during the second semester of a three-year undergraduate program in computer science at a small university with an approximate total student population of three thousand. The computer science program within this university consists of approximately two hundred students. This course serves as a continuation of the initial programming course undertaken by students in the preceding semester. The language used was Java. The GUI code was built with JavaFX: version 8 or newer was enough, but the recommended version was 17. The base code was scaffolding but was crucial to reducing code plagiarism from external sources. The assignment is suitable for any language with adequate GUI programming support.

4 STRENGTHS AND WEAKNESSES

We have identified several strengths in this assignment and two aspects deserving special attention. The identified strong points were the following:

(1) It is a simple graphical game that intellectually challenges the player.
(2) Provides good opportunities to discuss object-oriented design:
   (a) What are the objects?
   (b) What do those objects do? Explore similarities between the objects;
   (c) The keeper (pusher) and the boxes can have a supertype (a mobile object), and polymorphism can be explored;
   (d) A strong separation between interface code and model (game logic) code provides a good opportunity to stress the importance of encapsulation and reusability;
(3) It offers good motives to introduce worker threads (replaying the game), file processing, and GUI programming;
(4) The assignment complexity can easily be tailored to the course objectives and students' backgrounds using the provided base code, scaffolding, or both.
(5) The game can also be explored in artificial intelligence courses, especially machine learning (e.g., [8]).

The following points deserve some attention:

(1) Attractive solutions demand multiple classes, which can easily overwhelm some students; a base code with the base model-view architecture can be used, containing at least one example of the two-way communication between the GUI code and the game logic (model) code;
(2) Sokoban code on the internet can be a problem; the solution was forcing the use of a specific code base as a starting point. Also, the handout demands specific classes;
(3) Students must know relatively well one programming language, and at least one previous programming course is advisable.
(4) A language or a framework with simple support for GUI programming is highly recommended.

We stayed away from more advanced requirements to which Sokoban is also very amenable, namely the creation of a solver (e.g., [5]) or puzzle generation (e.g., [3]). As a more advanced requirement, we asked students to replay the game, which can also be a stored solution. However, the objective was to use a worker thread in the form available in the GUI framework, not to find the solution automatically. All the work (thinking) was always on the player's side.

5 POSSIBLE VARIANTS

It is effortless to think of variants. For example, (1) more than one player alternating or collaborating (playing concurrently); (2) "circular" board where the keeper can leave from one side and enter in the opposite side; (3) undo/redo; (4) replay of the game which can be used to show live resolutions; (5) use of mouse, keyboard, or both; (6) the additions of nice sounds; (5) skins with several different images, colors, or both.

6 RESULTS AND CONCLUSIONS

The informal feedback provided by students was overwhelmingly positive. The students appreciated the game's accessibility, as it could easily be played. The assignment's difficulty level facilitated the attainment of a basic, minimal grade while presenting a challenge for those aiming for higher grades. Some students did not know the game but developed an affinity for it afterward. The users also expressed appreciation for the game's logical clarity, consequently facilitating a more streamlined understanding of the requirements.

ACKNOWLEDGMENTS

This research was funded (in part) by the Portuguese FCT program, Center of Technology and Systems (CTS) UIDB/00066/2020 / UIDP/00066/2020.

REFERENCES

[1] Robert Ball, Linda DuHadway, Spencer Hilton, and Brian Rague. 2018. GUI-Based vs. Text-Based Assignments in CS1. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 1017–1022. https://doi.org/10.1145/3159450.3159463
[2] Jessica D. Bayliss and Sean Strout. 2006. Games as a "Flavor" of CS1. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (Houston, Texas, USA) *(SIGCSE '06)*. Association for Computing Machinery, New York, NY, USA, 500–504. https://doi.org/10.1145/1121341.1121498
[3] Barbara De Kegel and Mads Haahr. 2020. Procedural Puzzle Generation: A Survey. *IEEE Transactions on Games* 12, 1 (2020), 21–40. https://doi.org/10.1109/TG.2019.2917792
[4] Dorit Dor and Uri Zwick. 1999. SOKOBAN and other motion planning problems. *Computational Geometry* 13, 4 (1999), 215–228. https://doi.org/10.1016/S0925-7721(99)00017-6
[5] Zheng Li, Liam O'Brien, Shayne Flint, and Ramesh Sankaranarayana. 2014. Object-oriented Sokoban solver: A serious game project for OOAD and AI education. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. 1–4. https://doi.org/10.1109/FIE.2014.7044115
[6] Sokoban. 2022. sokoban.info. https://sokoban.info/ [Online; accessed 13-October-2022].
[7] Wikipedia contributors. 2022. Sokoban — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Sokoban&oldid=1100751512 [Online; accessed 13-October-2022].
[8] Zhao Yang, Mike Preuss, and Aske Plaat. 2022. Transfer Learning and Curriculum Learning in Sokoban. In *Artificial Intelligence and Machine Learning*, Luis A. Leiva, Cédric Pruski, Réka Markovich, Amro Najjar, and Christoph Schommer (Eds.). Springer International Publishing, Cham, 187–200.

Special Education Teachers Evaluating the Accessibility of CS Educational Robotics

Andrew Bennett
University of Florida
bennettandrew@@ufl.edu

Maya Israel
University of Florida
misrael@coe.ufl.edu

Joanne Barrett
University of Florida
jrbarrett@coe.ufl.edu

Debra "Kelly" Thomas
Broward County, FL Public Schools
kelly.thomas@browardschools.com

Jason McKenna
VEX Robotics
jason_mckenna@innovationfirst.com

ABSTRACT

All students benefit when computer science (CS) materials are accessible, but it is critical for students with disabilities. In order to provide opportunities for all students to be successful, it is important for teachers to be able to evaluate the accessibility of their lessons and technology. One way to evaluate accessibility is the POUR framework. The POUR framework represents what can be Perceived through the senses, how users can Operate a material or technology, how it is Understandable to users, and the overall Robustness. POUR provides a promising way for K-12 CS teachers to evaluate accessibility for their learners. We describe how the POUR framework was used by a cohort of teachers to evaluate VEX 123 for their learners with disabilities. Findings from the teacher POUR analysis revealed that overall, the teachers noted that the VEX 123 provided the necessary range of entryways into coding through its three modalities: The touch coding on the robot itself, the coder cards, and VECcode (the block-based coding environment). At the same time, the teachers indicated that some students with disabilities faced a number of motor and sensory difficulties. Overall, this study showcased a way for teachers to provide insight into the level of accessibility of CS education tools specific to their students' strengths and needs.

KEYWORDS

Teacher Education, Computer Science Education, Teachers' Identity, Teachers' Values

ACM Reference Format:

Andrew Bennett, Maya Israel, Joanne Barrett, Debra "Kelly" Thomas, and Jason McKenna. 2024. Special Education Teachers Evaluating the Accessibility of CS Educational Robotics. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635576

1 INTRODUCTION

One way CS education can be supported is through the use of accessible materials. It has been suggested that accessibility can be better maximized if accessibility is considered at the start of the design or evaluation process, not as an afterthought [8]. However, with many CS education tools, this is not always possible as these were not designed with accessibility in mind [**?**].

Teachers are essential to understanding how educational products have been used or not used in the classroom as they routinely interact with the technologies, use them with students, and evaluate its usage [1]. Hence, partnering with teachers to provide feedback on the accessibility of CS education tools and robotics can help to ensure that CS instruction is inclusive of all students.

One way for teachers to evaluate accessibility is through using the Perceivable, Operable, Understandable, and Robust (POUR) guidelines [4]. POUR was created to streamline accessibility guidelines for websites [5]. It now can be used to evaluate accessibility, including educational materials [9]. Yet, little research has applied the POUR principles to CS education.

We picked VEX 123 as a product to examine due to commitment from VEX to make their technologies more accessible. Additionally, educational robotics are important for increasing student learning, motivation, and semse pf well-being [2]. VEX 123 is a hybrid educational robot aimed at pre-kindergarten through early elementary students; generally it is an entry point of CS education for the youngest learners [7]. VEX 123 is distinguished by its three ways to program: directly manipulating touch buttons on the device, tangible coding through a blue-tooth enabled coder and code card system, and programming done on the VEXCode online coding platform [10].



Figure 1: Three ways of coding with VEX 123

We created a professional development module for special education teachers on using the POUR framework to evaluate the CS education tools that they use with learners with disabilities. This module is part of a larger project funded by Google aimed at wide-scale professional development focused on computer science inclusion and accessibility. As part of the module, the teachers were asked to examine a technology they used in the classroom as well as taking part in a group discussion of VEX 123 and POUR. This

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635576

poster presents the findings of the teachers' POUR analysis of the VEX 123.

2 TEACHER POUR ANALYSIS OF ROBOTICS

Data that was used as part of our understanding of how teachers used the POUR framework were: (1) Notes from meetings with the teachers as they described their experiences with the VEX123, and (2) materials that teachers created within their professional development course management system (CMS). These sources of data were deductively analyzed for themes within each of the POUR principles. POUR analysis data was classified according to concerns teachers brought up. Data from the meetings was categorized based on particular sub-questions for each POUR principle.

2.1 Perceivable

Teachers felt the three ways to code was useful and that the pictures were helpful to younger, preliterate children. One concern was that even though there was audible feedback, it was not necessarily loud enough in order to be heard; adding a bluetooth speaker was suggested. Another concern involved students having motor difficulties in particular with regards to getting coder cards in and out of the coder. Teachers also believed that braille for the coder cards would be helpful for students with visual impairments.

2.2 Operable

Like with perceivability, the teachers acknowledged that the touch-coding on the VEX 123 robot as well as the use of the coder cards allowed most of their students to engage in CS in positive ways. At the same time, teachers thought the coder cards might be problematic for students with some motor disabilities. Some were also concerned that some students might not be able to use the touch buttons and drag and drop independently.

2.3 Understandable

Teachers felt that VEX 123 was very understandable, especially the touch-coding robot. In fact, the physical output of the robot helped reinforce concepts that were otherwise generally abstract. Teachers did note different levels of complexity between the three ways to code, which allowed students with different experiences and levels of understanding to all engage with the computational activities. One suggestion that teachers made was to incorporate strategies for students to ask for help when they struggled with understanding computational concepts. For example, they suggested making "help" cards with messages about the level of help needed.

2.4 Robust

Teachers indicated that they struggled with making VEX 123 compatible with assistive technologies such as eye trackers and switches. This challenge sometimes involved understanding how the assistive technologies interfaced with the VEX 123 and sometimes was an issue of true compatibility. The pairing of devices could also be lost very quickly. Teachers had mixed opinions on durability with one saying it was durable when students dropped the device with another discussing the fragility of the board and strips.

3 DISCUSSION

There is growing research suggesting that educational robotics can support learners' understanding of computational thinking in engaging ways [7]. However, many of these studies have not focused on the inclusion of learners with disabilities [6], [**?**]. This study demonstrates POUR as a potential framework to address accessibility.

Although there are ways we could improve the accessibility module presented to the teachers, this study added to ways teachers can evaluate educational technologies used to teach computer science concepts with accessibility in mind. As demonstrated here, teachers voiced a variety of positives and areas of improvement. Analysis was tempered by the needs of teachers' students.

While we have concentrated on VEX 123 in this study, the POUR method is technology agnostic. Future work will incorporate a greater range of educational technologies used to teach programming. We also still do not have a deep understanding of how teachers may evaluate accessibility together or what students think of these technologies which form limitations of this work. Dyadic interviewing may form new perspective.s [3].

ACKNOWLEDGMENTS

We are grateful to the teachers who participated in the Inclusion in Computer Science Workshop. This project was funded by Google and Innovation First International dba VEX Robotics.

REFERENCES

[1] Christina Chalmers. 2018. Robotics and computational thinking in primary school. *International Journal of Child-Computer Interaction* 17 (2018), 93–100.
[2] Linda Daniela and D. Lytras, Miltiadis. 2019. Educational Robots for Inclusive Education. *Technology, Knowledge, and Learning* 24 (2019), 93–100.
[3] Paulina Haduong and Karen Brennan. 2020. Talking in Pairs: Learning From and With Teachers Through Artifact-Based Dyadic Interviews. In *Proceedings of the 14th International Conference of the Learning Sciences*. 2369–2370.
[4] Sarah Horton and Whitney Quesenbery. 2014. *A web for everyone: Designing accessible user experiences*. Rosenfeld Media.
[5] W3C Website Accessibility Initative. 2021. *Module 4: Principles, Standards, and Checks*. https://www.w3.org/WAI/curricula/foundation-modules/principles-standards-and-checks/
[6] Maya Israel, Shireen Hafeez, Emmanuel Schanzer, Rebecca Dovi, Emma Koslow, and Todd Lash. 2019. Panel: Making K-12 CS Education Accessibility a Norm, not an Exception. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 1232–1233.
[7] Yuhan Lin. 2022. Exploring the Child-Robot Interaction with the Programming in Mind: Bridging Physical and Virtual Programming for Young Children. In *Interaction Design and Children*. 683–686.
[8] CAST Luis Pérez. 2022. Bake the blueberries in: Designing for accessibility with POUR. https://aem.cast.org/get-started/events/2022/10/bake-the-blueberries-in-designing-for-accessibility-with-pour
[9] National Center on Accessible Educational Materials (NCAEM). nd. Designing for Accessibility. https://aem.cast.org/create/designing-accessibility-pour
[10] VEX Robotics. [n. d.]. *VEX 123*. https://vexrobotics.com/123

Supporting Mastery Learning Through an Adaptive Grade Portal

Connor Robert Bernard
connorbernard@berkeley.edu
University of California, Berkeley
Berkeley, CA, USA

Dev Ahluwalia
dahluwalia@berkeley.edu
University of California, Berkeley
Berkeley, CA, USA

Madelen Flores
madelenf143@berkeley.edu
University of California, Berkeley
Berkeley, CA, USA

Wilson Chu
wilsnchu@berkeley.edu
University of California, Berkeley
Berkeley, CA, USA

Yuanhan Li
williamlyh@berkeley.edu
University of California, Berkeley
Berkeley, CA, USA

Daniel Garcia
ddgarcia@cs.berkeley.edu
University of California, Berkeley
Berkeley, CA, USA

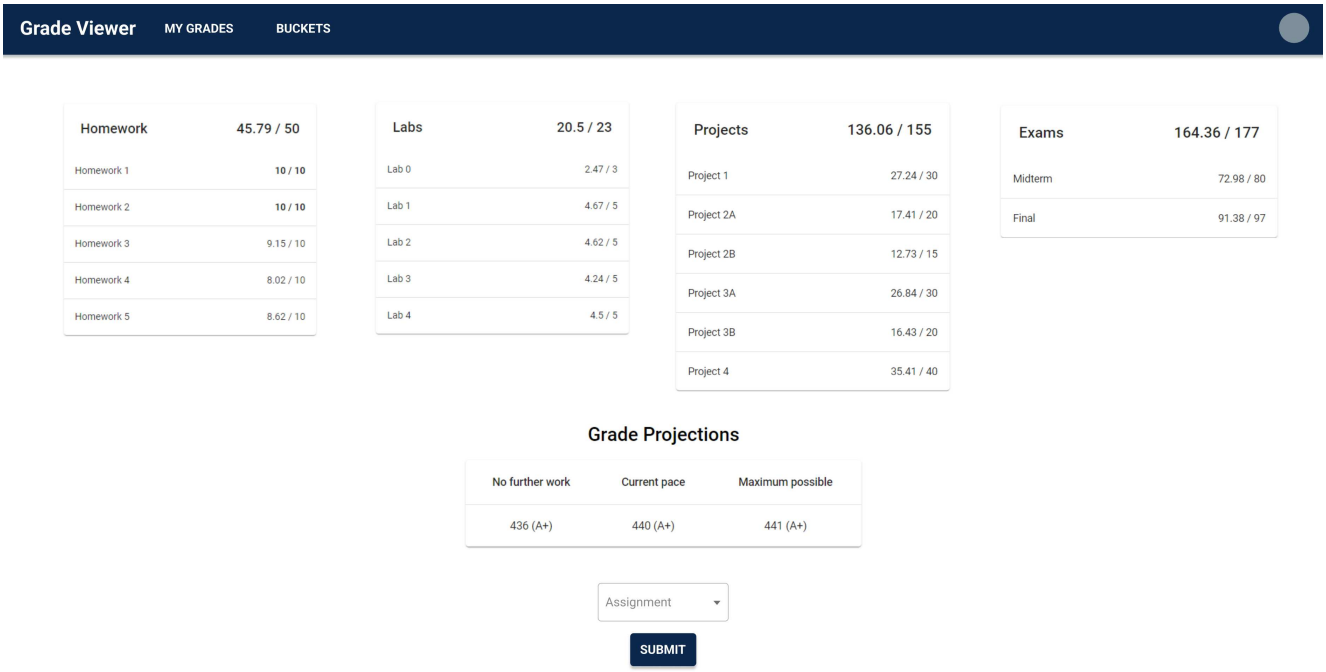


Figure 1: The dashboard seeks to provide students a streamlined UI to stay up-to-date with their grades and submit assignments.

ABSTRACT

Gradeview is an innovative and ongoing project presenting a comprehensive dashboard designed to cater to the needs of both students and educators, offering a centralized platform for monitoring and managing course progress. This work in progress draws its data from a single Google Sheet ensuring that updates, such as new grades and categories, are seamlessly reflected across all student and administrator dashboards instantaneously. *Gradeview* leverages a plethora of other technologies that can drastically increase both student and educator productivity beyond the classroom setting. For instance, it introduces an interactive mastery learning *Concept Map* customized for each student. The organization structure of *Gradeview* promotes completion of work regardless of timeline so as to benefit student mental health and increase mastery of core concepts in the class. Our *Director* module enables the automated management of students' grades for those students working past the term deadline. In order to facilitate the implementation of these new dashboard modules, we developed the project to be an easy-to-use microservice architecture that can automatically track individual and overall student performance through course grading metrics.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635621

KEYWORDS

Dashboard, Mastery Learning, Instructor Tools, Education, Software

ACM Reference Format:
Connor Robert Bernard, Dev Ahluwalia, Madelen Flores, Wilson Chu, Yuanhan Li, and Daniel Garcia. 2024. Supporting Mastery Learning Through an Adaptive Grade Portal. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635621

1 BACKGROUND

Dashboards showing student progress are not novel, every learning management system (e.g., Canvas) has that feature, usually based on a weighted sum of assessment items, like homework, exams, and projects. What makes *Gradeview* unique is its ability to work flexibly with any custom grading policy that an instructor utilizing mastery learning wishes to employ. Other developers have built systems that feature a "learning dashboard"; Khan Academy even shows a student recommended lessons and practice problems [1]. Further, it includes a breakdown of how close a student is to mastery with each topic, and provides visualizations to help students understand this data. A plug-in was developed to give students more high-level analytics, including visualizations and recommendations, and for professors to see a suite of visualizations for each student [2]. Some dashboards even claimed they could predict if a student would pass a course [3], based on past performance.

Our research aims to build on this previous work, and support all the elements needed for mastery learning: grade feedback (and support complicated grading formulas), visualization of where they are in the overall course topic Concept Map, and the ability for a student to submit work asynchronously.

We hope to address the following research questions:

(1) Can a student portal centered around maintaining opportunity amidst hardships encourage students to continue learning in a healthier way (mentally)?
(2) Can visualizations for students allow them to effectively measure which course topics are most challenging and seek to remediate those?
(3) How can autonomous registrar tools encourage instructors to develop more accommodating grading policies?

2 GRADEVIEW MODULES

These are the two custom modules developed specifically for the *Gradeview* dashboard, which shows a view into the grade book as shown in Figure 1.

2.1 Concept Map

Concept maps have long been used in education to visualize knowledge topics and their interdependencies. Many applications exist to create them interactively, but tools for creating them programmatically often have steep learning curves or arcane syntax. Worse yet, they are typically static and the content can not be easily personalized for each student. The concept map in *Gradeview* is *generated from a plain-text source file and automatically colored based on grade and assignment data*, and it allows students to visualize their progress in the class dynamically, as in Figure 2. Nodes are color-coded: node fill colors represents individual students' mastery of the concept while border colors shows the staff-defined progress of the class. If desired, students could have downstream topics "locked" until they master all the earlier, prerequisite ones. It also allows students to compare their progress against the course schedule and identify areas for improvement. Clicking on the nodes presents them with staff-provided resources for corresponding learning modules.

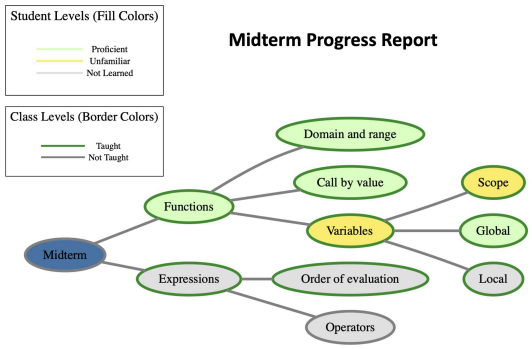


Figure 2: *Concept Map* shows student's progress on each topic in the class.

2.2 Director

Director is a software system that gives students the opportunity to submit coursework past the end-of-semester deadline. Traditionally, once this work is completed, the grade is entered into the previous term's grade book manually. *Director* allows the gradebook to be updated automatically, with no instructor intervention.

This requires interaction with three main systems: an assessment platform (where assignments are hosted), a grade database (where student scores are recorded) and a registrar (student information system). Once integrated with these, *Director* can update scores for resubmitted work and even submit the updated grade to the registrar.

The usage of *Director* coupled with *Gradeview* benefits the mental health of students by allowing them to continue working on projects past the course end date. These tools can help educators create a more accommodating space for students, without requiring additional manual effort from educators.

REFERENCES

[1] Alex Rayón, Mariluz Guenaga, and Asier Núñez. 2014. Integrating and Visualizing Learner and Social Data to Elicit Higher-Order Indicators in SCALA Dashboard. In *Proceedings of the 14th International Conference on Knowledge Technologies and Data-Driven Business* (Graz, Austria) *(i-KNOW '14)*. Association for Computing Machinery, New York, NY, USA, Article 28, 4 pages. https://doi.org/10.1145/2637748.2638435
[2] José A. Ruipérez-Valiente, Pedro J. Muñoz Merino, and Carlos Delgado Kloos. 2013. An Architecture for Extending the Learning Analytics Support in the Khan Academy Framework. In *Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality* (Salamanca, Spain) *(TEEM '13)*. Association for Computing Machinery, New York, NY, USA, 277–284. https://doi.org/10.1145/2536536.2536578
[3] Candace Thille. 2014. MOOCs and Technology to Advance Learning and Learning Research Opening Statement: MOOCs and Technology to Advance Learning and Learning Research (Ubiquity Symposium). *Ubiquity* 2014, April, Article 1 (apr 2014), 7 pages. https://doi.org/10.1145/2601337

Co-Designing Integrated CS Curriculum Artifacts with K-5 Classroom Teachers

Maverick Berner
maverick.berner@marquette.edu
Marquette University
Milwaukee, WI, USA

Maximus Berner
maximus.berner@snc.edu
Marquette University
Milwaukee, WI, USA

Darcy Ronan
ronand@sacredheart.edu
Sacred Heart University
Fairfield, CT, USA

Dennis Brylow
dennis.brylow@marquette.edu
Marquette University
Milwaukee, WI, USA

ABSTRACT

What would happen if generalist K-5 teachers had the technical support to build custom digital tools integrating computing concepts across other fields of study? This could open new avenues for students to learn computational thinking, without compromising time spent in other important subject areas. Furthermore, it could grant teachers the same level of creativity in developing computing resources that they have in other curriculum areas, rather than being limited by available resources or by their own lack of expertise. With this poster, we will cover our approach to co-designing digital tools with K-5 teachers for the primary goal of integrating computing concepts into a variety of other subject areas. We reflect on the observations we made throughout the collaboration and how they impacted our approach moving forward. Additionally, we will highlight some of the tools that came out of the collaboration and how co-designing a specialized digital tool from the ground up granted teachers the flexibility to explore ideas beyond integrating computer science concepts, including student accessibility, engagement, and social emotional learning (SEL).

ACM Reference Format:
Maverick Berner, Maximus Berner, Darcy Ronan, and Dennis Brylow. 2024. Co-Designing Integrated CS Curriculum Artifacts with K-5 Classroom Teachers. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635617

1 INTRODUCTION

As technological innovations continue to advance all fields forward, computer science (CS) is increasingly considered an essential subject. Consequently, there is a growing nationwide acknowledgement of the imperative to enhance K-12 CS education to meet public demand, embraced by the US Department of Education, researchers, and professionals alike [4]. Compared to other sciences, CS is still a relatively new and rapidly evolving field. While there are many significant challenges in advancing CS education, especially at the elementary level, a primary obstacle arises from the fact that many current educators and administrators were not exposed to CS during their own education. This lack of exposure results in reduced preparedness, confidence, and background knowledge to teach the subject effectively. A proposed solution to address the lack of space in curriculum is embedding CS concepts within an existing unit of another subject area. In doing so, these embedded lessons could take on a multifaceted approach: teaching both CS and the original topic within one unit. This integration has the potential to bolster students' ability to problem-solve and foster creativity [3, 5, 6]. An important aspect of this work was empowering active teachers to lead the development of these CS-Integrated units, with the support of developers for assistance in the production of digital tools for their curriculum design needs. By empowering educators and involving them in each stage of the software design process, we not only create impactful tools for the next generation of elementary students, but also foster confidence, promote exposure, and increase the background knowledge in CS for the teachers themselves.

2 PROJECT CONTEXT

The design and implementation of novel CS-embedded curriculum units is the culmination of teachers' participation in *Project {FUTURE}*, a multi-year, multi-state engagement between CS education researchers and elementary teachers. The project provided introductory CS professional learning to all participating elementary teachers (N = 791) at partner schools (N = 28) via Code.org CS Fundamentals [1] workshops ("CSF Intro"). Subsequent supports included the CSF "Deeper Dive" [2] as well as an instructional coaching program. Teachers were then recruited for a two-year unit development and testing process. Teachers who participated in this smaller and more intensive experience would ultimately gather for a five-day culminating summer institute, the goal of which was to realize the co-design of novel digital tools to support CS-embedded units. To achieve this goal, the summer institute featured an intensive collaboration between the teachers, with their expertise in curriculum design, and a team of CS content experts and software developers, capable of building a digital tool to the teachers' specifications. The Collaborative was launched in Wisconsin (N = 12) and Connecticut (N = 26).

The teachers were assembled into professional learning communities ("PLCs"), typically comprised of teachers from the same district or with a similar topic or standard, to design one CS-integrated

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635617

unit involving a digital tool or component. Throughout the week of the institute, the teachers would rotate between formal professional development modules, flexible time to work toward conceptualizing their CS-integrated unit, and consultation with the team of developers about their digital component. A team of education and CS faculty and facilitators from both host universities oversaw the overall week of professional development.

Our team broke the co-design work of the summer institute into six phases: *1. Conceptualizing*: Working from their overall unit design, each PLC conceptualizes a digital tool to provide for an integrated and authentic application of CS and target content concepts for students. *2. Discussion*: The PLC groups outline the core idea and key details of their conceptualized digital tool to the developer team in breakout sessions. If necessary, the developers contribute their own background expertise to help draw out where and how CS learning can be expressed further. *3. Research*: The developer team conducts independent research surrounding related work and technological considerations for designing to the teachers specifications. *4. Prototyping and Continued Discussion*: The developers present a prototype of the digital tool to the teachers. Teachers provide their initial feedback on the digital tool. *5. Polishing*: Using the feedback provided by teachers, the developers further refine the tool and make adjustments where necessary. *6. Professionalizing*: The developers make final considerations to improve sustainability, accessibility, and scalability, to promote the tool's widespread adoption.

3 DEVELOPMENT TEAM

For the 2023 summer institutes, the development team was comprised of 8 undergraduate and 2 graduate student researchers, plus a faculty mentor. For the first summer institute in Wisconsin, the entire development team participated in the breakout session discussions with the teacher PLCs. Due to the distance (1000 miles) between the two states, the second institute split the development team into an *away team* of 6 students plus the faculty mentor who would lead the PLC meetings on site, and a *home team* who provided remote research and technical support focused on prototype development.

To keep the discussions between the development team and PLC groups on track, a template was created to guide the conversation, as well as prevent crucial details from being missed. The template was split into three sections: "Hardware Constraints", "Digital Tool Description", and "Storyboarding". It was expected that many teachers might not be aware of the wide breadth of existing CS education curricula and digital education tools for K-5 classrooms beyond those presented in their introductory workshops. To give the teachers a sense of what was already available, each member of the development team selected a popular computing or robotics education tool and prepared a demo to present to the teachers. If PLC groups wanted to design their CS-integrated unit around pre-existing tools, they were encouraged and supported in exploring those options.

4 SUMMARY AND OBSERVATIONS

By the final day of both institutes, the prepared prototypes were presented to the teachers. Following the presentations, each PLC group met once more with the development team to provide valuable feedback on their early first impressions. Responses from the PLC groups were overwhelmingly positive, with some groups even suggesting they could implement the tool as is.

This poster highlights key factors contributing to the success of the institutes. We will discuss the adjustments made while transitioning between the two institutes, the motivations behind these changes, and how they improved the results. One such change involved reducing the number of developers in the discussion rooms with the PLC groups, which streamlined communication, improved the natural flow of the conversation, and allowed teachers more uninterrupted time to express their ideas clearly.

5 CONCLUSION AND FUTURE WORK

In total, the PLC groups designed 17 novel CS-integrated units across the two summer institutes. Of these, five required custom software to be developed; the remainder were designed around an existing CS education tool, a handful requiring additional support from the development team. A record of each project and its requirements was created, and each member of the development team was assigned one project to deliver. All of the projects are publicly available via Project {FUTURE} host site.

Putting teachers at the forefront of designing these units and tools realized the potential impact this collaboration can have in the classroom. Not only can these tools be used to integrate computing concepts across all areas of study, but many accomplish this while also improving other aspects of the student learning experience. Teachers employed numerous and varied strategies to promote student accessibility and enrich lessons with additional concepts including aspects of social emotional learning. The tools produced thus far could only have been possible through this collaboration as neither the development team, nor the teachers, wielded both the knowledge of curriculum design and technical expertise to create them independently. The Project {FUTURE} team will continue develop these tools as they are piloted in the classroom, then the teachers and development team will regroup to assess the project's success, and how the tools could be refined even further. We then plan to consider our observations throughout the entire process to improve our approach toward a second collaboration.

REFERENCES

[1] Code.org. 2023. CS Fundamentals. https://code.org/educate/csf
[2] Code.org. 2023. CS Fundamentals Deep Dive Workshop. https://code.org/professional-development-workshops
[3] K. Fisler, E. Schanzer, S. Weimar, A. Fetter, K. Renninger, S. Krishnamurthi, J. Politz, B. Lerner, J. Poole, and C. Koerner. 2021. Evolving a K-12 Curriculum for Integrating CS into Mathematics. In *Proceedings of SIGCSE '21* (Virtual Event, USA) . ACM, 59–65. https://doi.org/10.1145/3408877.3432546
[4] U.S. Department of Education. 2016. Announcing the CS for All Initiative. 6 (02 2016). https://sites.ed.gov/whhbcu/2016/02/05/announcing-the-computer-science-for-all-initiative/
[5] S. Rodger, M. Dalis, C. Gadwal, J. Hayes, P. Li, F. Wolfe, W. Zhang, and L. Liang. 2012. Integrating computing into middle school disciplines through projects. In *Proceedings of SIGCSE '12*. 421–426.
[6] D. Yang, Y. Baek, Y. Ching, S. Swanson, B. Chittoori, and S. Wang. 2021. Infusing Computational Thinking in an Integrated STEM Curriculum: User Reactions and Lessons Learned. *Euro. J. of STEM Ed.* 6, 1 (2021), 4. https://doi.org/10.20897/ejsteme/9560

Using Notional Machines to Automatically Assess Students' Comprehension of Their Own Code

Joey Bevilacqua
joey.bevilacqua@usi.ch
Software Institute, Università della Svizzera italiana
Lugano, Switzerland

Luca Chiodini
luca.chiodini@usi.ch
Software Institute, Università della Svizzera italiana
Lugano, Switzerland

Igor Moreno Santos
igor.moreno.santos@usi.ch
Software Institute, Università della Svizzera italiana
Lugano, Switzerland

Matthias Hauswirth
matthias.hauswirth@usi.ch
Software Institute, Università della Svizzera italiana
Lugano, Switzerland

ABSTRACT

Code comprehension has been shown to be challenging and important for a positive learning outcome. Students don't always understand the code they write. This has been exacerbated by the advent of large language models that automatically generate code that may or may not be correct. Now students don't just have to understand their own code, but they have to be able to critically analyze automatically generated code as well.

To help students with code comprehension, instructors often use notional machines. Notional machines are used not only by instructors to explain code, but also in activities or exam questions given to students. Traditionally, these questions involve code that was not written by students. However, asking questions to students about their own code (Questions on Learners' Code, QLCs) has been shown to strengthen their code comprehension.

This poster presents an approach to combine notional machines and QLCs to automatically generate personalized questions about learners' code based on notional machines. Our aim is to understand whether notional machine–based QLCs are effective. We conducted a pilot study with 67 students to test our approach, and we plan to conduct a comprehensive empirical evaluation to study its effectiveness.

ACM Reference Format:
Joey Bevilacqua, Luca Chiodini, Igor Moreno Santos, and Matthias Hauswirth. 2024. Using Notional Machines to Automatically Assess Students' Comprehension of Their Own Code. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635524

1 INTRODUCTION

Code comprehension is an important skill to acquire and is considered important for a positive learning outcome [3, 9, 10]. Learners often struggle to understand their code, which may be a mix of their own production, code from textbooks, and code found online. With the increasing capabilities and popularity of large language models, which are able to generate code from a user–supplied prompt, this problem becomes more and more relevant. The bar for *producing code* gets lowered even more. Empowering learners with such tools might have the consequence of decreasing the comprehension of their own code, given that there is less need to reason about the various aspects that pertain to the implementation. Moreover, these models provide no guarantee that the generated code is correct: code comprehension as a skill has never been more crucial.

To help students with code comprehension, instructors often use *Notional Machines* [11], "pedagogic devices used to assist the understanding of some aspect of programs or programming" [4]. Notional machines are used not only by instructors to explain code, but also in activities or exams given to students. Traditionally, the code used in these tasks is not student code. However, asking questions to students about *their own* code has been shown to strengthen their code comprehension [5]. Recent work on *Questions on Learners' Code* (QLCs) [7] explored approaches to automatically generate questions about the code written by students. QLCs are usually multiple–choice questions and can thus be easily graded automatically. Multiple studies [6, 8] have shown that bad scores on QLCs correlate with bad performance in courses.

Our approach combines QLCs and notional machines to automatically generate and assess notional machine–based questions about learners' code. Our aim is to understand the effectiveness of notional machine–based QLCs.

2 INSTANTIATING THE APPROACH

To study our approach we need to focus on a specific notional machine. We picked *Expression as Tree* [4] because expressions have been shown to be prevalent in the largest repository of student code (Blackbox [1]) but neglected in textbooks [2].

We implemented a platform that instantiates our approach for teaching expressions in Java using the *Expression as Tree* notional machine. In our platform, the instructor selects expression–related constructs such as arithmetic operators, method invocations, and array accesses by specifying a query in a domain–specific language. Using the code already written by learners in their programming assignments, our platform finds expressions that include the instructor–specified constructs. Based on those expressions, it generates personalized tasks for students to solve. Specifically, students are

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635524

asked to translate source code into expression trees and to specify the types of sub–expressions.

```
class LinAlgebra {
  double[] scale(double a, double[] x) { ... }
  double scaledAt(double a, double[] x,
                  int oneBasedPos) {
    return scale(a, x)[oneBasedPos - 1];
  }
}
```

Listing 1: An expression (yellow) containing a method invocation and an array access, selected from student code.

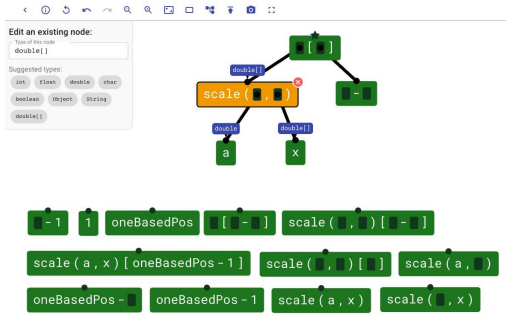


Figure 1: An expression tree (top) being constructed using automatically generated nodes (bottom) which include distractors. Nodes can be annotated (left) with types (blue labels).

Figure 1 illustrates an activity automatically generated from the expression highlighted in Listing 1. The task of the student is to construct the tree corresponding to the given expression. They are provided with a set of disconnected nodes. Some nodes are necessary for constructing the correct tree, other nodes are distractors our platform automatically generates. Students also need to specify the type of each sub-expression by attaching labels to each node.

The platform also generates the reference solution: the correct expression tree with the correct types specified for all nodes. It compares the reference solution with the student's submission and produces feedback. The feedback for students highlights incorrectly represented constructs and reports errors in the tree structure. The feedback for the instructor includes aggregated results and summarizes the observed mistakes tallying incorrectly represented constructs.

3 PILOT STUDY AND FUTURE WORK

We performed a pilot study in a university–level Java programming course with 93 students, of which 67 gave consent for their data to be included in our study. We used the platform for two programming assignments. For each assignment, each student received a personalized task, automatically constructed from the code they submitted as a solution to the assignment. Solving these tasks was not mandatory for students; in the end, we received a total of 52 solutions to the notional machine-based questions. The platform identified 25 incorrect tree nodes, which it reported in feedback for the students and the course instructor.

Our preliminary results show that our approach is practical, with the tool being used in classes of a university–level course on object–oriented programming. Our next step is to conduct a comprehensive empirical evaluation. Our evaluation will include the quality of the automatically generated distractor nodes, the effectiveness, and the accuracy of the automatic feedback generation for both instructors and students, as well as the correlation of the performance of learners on the tasks with their performance in other assessments.

ACKNOWLEDGMENTS

This work was partially funded by the Swiss National Science Foundation project 200021_184689.

REFERENCES

[1] Neil Christopher Charles Brown, Michael Kölling, Davin McCall, and Ian Utting. 2014. Blackbox: A Large Scale Repository of Novice Programmers' Activity. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 223–228. https://doi.org/10.1145/2538862.2538924

[2] Luca Chiodini, Igor Moreno Santos, and Matthias Hauswirth. 2022. Expressions in Java: Essential, Prevalent, Neglected?. In *Proceedings of the 2022 ACM SIGPLAN International Symposium on SPLASH-E (SPLASH-E 2022)*. Association for Computing Machinery, New York, NY, USA, 41–51. https://doi.org/10.1145/3563767.3568131

[3] Peter Donaldson and Quintin Cutts. 2018. Flexible Low-Cost Activities to Develop Novice Code Comprehension Skills in Schools. In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education (WiPSCE '18)*. Association for Computing Machinery, New York, NY, USA, 1–4. https://doi.org/10.1145/3265757.3265776

[4] Sally Fincher, Johan Jeuring, Craig S. Miller, Peter Donaldson, Benedict du Boulay, Matthias Hauswirth, Arto Hellas, Felienne Hermans, Colleen Lewis, Andreas Mühling, Janice L. Pearce, and Andrew Petersen. 2020. Notional Machines in Computing Education: The Education of Attention. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '20)*. Association for Computing Machinery, New York, NY, USA, 21–50. https://doi.org/10.1145/3437800.3439202

[5] Rita Garcia, Katrina Falkner, and Rebecca Vivian. 2019. Instructional Framework for CS1 Question Activities. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '19)*. Association for Computing Machinery, New York, NY, USA, 189–195. https://doi.org/10.1145/3304221.3319732

[6] Teemu Lehtinen, Lassi Haaranen, and Juho Leinonen. 2023. Automated Questionnaires About Students' JavaScript Programs: Towards Gauging Novice Programming Processes. In *Australasian Computing Education Conference*. ACM, Melbourne VIC Australia, 49–58. https://doi.org/10.1145/3576123.3576129

[7] Teemu Lehtinen, Andre L. Santos, and Juha Sorva. 2021. Let's Ask Students About Their Programs, Automatically. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*. IEEE, Madrid, Spain, 467–475. https://doi.org/10.1109/ICPC52881.2021.00054

[8] Teemu Lehtinen, Otto Seppälä, and Ari Korhonen. 2023. Automated Questions About Learners' Own Code Help to Detect Fragile Prerequisite Knowledge. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. ACM, Turku Finland, 505–511. https://doi.org/10.1145/3587102.3588787

[9] Raymond Lister, Colin Fidge, and Donna Teague. 2009. Further Evidence of a Relationship between Explaining, Tracing and Writing Skills in Introductory Programming. *ACM SIGCSE Bulletin* 41, 3 (July 2009), 161–165. https://doi.org/10.1145/1595496.1562930

[10] Greg L. Nelson, Benjamin Xie, and Andrew J. Ko. 2017. Comprehension First: Evaluating a Novel Pedagogy and Tutoring System for Program Tracing in CS1. In *Proceedings of the 2017 ACM Conference on International Computing Education Research - ICER '17*. ACM Press, Tacoma, Washington, USA, 2–11. https://doi.org/10.1145/3105726.3106178

[11] Juha Sorva. 2013. Notional Machines and Introductory Programming Education. *ACM Transactions on Computing Education* 13, 2 (June 2013), 1–31. https://doi.org/10.1145/2483710.2483713

My Learnings from Allowing Large Language Models in Introductory Computer Science Classes

Rasika Bhalerao, Ph.D.
r.bhalerao@northeastern.edu
Northeastern University
United States

ABSTRACT

Many instructors want to allow their students to use large language models (LLMs) in their introductory computer science courses, but they first want to see other instructors' results from doing so before taking on the risk in their own courses. Presented here are the results from allowing students to use LLMs in the second course in a sequence of intensive introductory courses designed to prepare students with a non-computational background for entry into a masters' degree program. We allowed students to use the internet and LLMs (such as ChatGPT or Github Copilot) to help with assignments, with guidelines to avoid plagiarism and encourage learning. We then surveyed students to ask about how they used LLMs, whether they saw others cheating, how they generally used internet-based resources on assignments and exams, and their feedback on the policies. We found that students are overwhelmingly using LLMs (and the internet generally) to learn and code "better" rather than cheat. These results are intended to be a starting point to spark discussion on the adoption of new technologies in introductory computer science courses. The authors themselves will continue teaching courses with the policy that students should interact with an LLM the way they interact with a person: students are encouraged to discuss and collaborate with it, but copying code from it is considered plagiarism.

KEYWORDS

AI, plagiarism, assignments, students

ACM Reference Format:
Rasika Bhalerao, Ph.D.. 2024. My Learnings from Allowing Large Language Models in Introductory Computer Science Classes. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635511

1 FURTHER INFORMATION

1.1 Introduction and Background

New technology has a history of affecting education. Tools such as Wolfram Alpha (and calculators) forced math instructors to consider the goals of their classes. Increasingly accurate online language translation required language instructors to revaluate what they wanted their students to learn. As instructors were still adjusting policies regarding internet access, large language models (LLMs) such as ChatGPT and Github Copilot revolutionized the writing of code and the ways in which computer science can be taught.

Student use of AI tools that were designed without a focus on educational purposes is relatively new, despite AI having been used as an educational tool for about fifty years [1, 2, 4, 6, 7]. Goldstein and Miller [3] postulated that students are not uniformly able to determine when to seek and accept help, so an effective system for helping students should include a model of the student (to personalize guidance based on concepts and skills it believes the student knows) and a model of an expert human tutor. Students are now using LLMs such as ChatGPT, which do not have a model of a student or instructor; students are being forced to decide on their own when to seek help and to understand the balance between "cheating" and accepting needed help. In this preliminary study, to help students find this balance, we provided students with structured guidelines on using LLMs to help with assignments.

1.2 Method

We explained the policy to students in the Syllabus and assignment instructions, then surveyed the students afterwards.

1.2.1 Syllabus. We motivated the policy by predicting that pair-programming with LLMs will be "as common as Google searching bugs or discussing algorithms with a classmate."

We then stated the policy for collaboration with an LLM "such as ChatGPT or Github Copilot." We said, "for each homework assignment, I will list the things that you can use a model for. You may not use the model for any tasks other than what I list in the assignments. For example, I might say that you can ask the model to write code to get the sum of all numbers in a set." Students were required to cite LLMs the way they were required to cite people, e.g. "I used Github Copilot to create the list on line 9."

Reasoning for this strict policy was explained in terms relevant to these particular students: "It is in your best interest (to learn what you need for coding interviews) to use language models only for the small tasks that I list."

1.2.2 Assignment Instructions. This course had 6 substantial homework projects in Java. Each assignment's instructions had a section called "What you may use a language model for" with sentences like "You may use a language model to write up to 3 tests." or "You may use a language model to write getColumn() or getRow()" ("getter" methods that directly returned an attribute).

We also had weekly multiple-choice quizzes following a peer instruction-based [5] model. Students took each quiz twice: once

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635511

alone, and again with other classmates. Students were allowed to "collaborate" with an LLM for these summative quizzes.

We also had midterm and final exams; LLMs were not explicitly allowed, but the course was taught when popular search engines were implemented using LLMs. Students were allowed to read anything from the internet, but "communicating" was not allowed.

1.2.3 Survey. We surveyed students asking for feedback on the policies. There was no incentive for students to submit the feedback, except to improve the course. The instructions included "This form is optional and anonymous. **Please take care to avoid including information that could be used to identify someone.**" One question was: "Do you consent to your responses in this survey being used for research purposes? All information shared will remain anonymous, and your participation is voluntary. Your responses will be used to improve this [...] course whether or not you consent to your responses being shared for research." We allowed respondents to exclude individual question responses from research. The survey was approved by the appropriate ethics review board.

1.3 Discussion of Results

The survey had 18 respondents (a 32% response rate).

1.3.1 Which language model, if any, do you use for subproblems on homework assignments? 28% said they never used an LLM. 61% said they used ChatGPT. 11% said they tried ChatGPT, but it wasn't useful because, e.g., "I feel like AI doesn't understand what I am saying, and the way they explain things made me even more confused."

1.3.2 How, if at all, do you use the language model? Is it useful? Students found ChatGPT useful in helping them to learn to code. Students reported that they do not copy the code, but rather "Get the idea and rewrite the code." Students appreciated that "chatgpt provides annotations for the code. I find it more useful to write it on my own after reading it online." Some also used it to fix errors in their existing code: "I copy the error code and ask the explanation."

The model seems to offer help the way a TA might. Students said that using AI to learn syntax is "faster and detailed." One student said they use ChatGPT as "a 'substitute' TA to help explain concepts that I haven't fully understood from the lecture alone. It is very useful and has helped me learn at an accelerated pace, since I can get more of my questions answered."

One student wrote that LLMs are "helpful with giving answers to clarification questions" and that "when working on assignments over the weekends or at night, we can easily get simple answers from the models rather than waiting for professors/TAs to respond," allowing the student to "save time during meetings with professors/TAs, as I can directly ask about bigger concepts/implementations rather than having to ask the smaller questions."

It sounded like generating the entire solution was not helpful. Many students complained that "the language model generated a mess of code and didn't offer a correct answer" or that they "have to correct many errors." Rather, it was more useful for "writing repetitive codes, documentation and tests."

1.3.3 Have you noticed others using a language model to cheat? One respondent said "yes" and explained "I have not seen it with my own eyes but I have heard it talked about frequently." The authors did not detect the cheating.

1.3.4 Do you understand the course content (even if you don't need to write it all from scratch)? This question used a Likert scale from 1 (no understanding) to 5 (confident in understanding). All answers were either "5," "4," or "3" (50%, 39%, and 11%, respectively).

1.3.5 What, if anything, did you search for online during an exam? This question was multiple choice with the option to add a custom option. The most common answers were: 72% said they looked up syntax for something, and 33% Googled errors / warnings. 5% looked up a concept or procedure. 22% reported that they did the exam without needing to look anything up.

1.3.6 During exams, should we allow writing on the internet, or is it enough that you can read from the internet? This was also multiple choice with the option to write something in. 83% responded saying that reading from the internet is enough. 17% selected "We should allow writing on the internet."

1.3.7 Have you noticed others using the internet to cheat on exams? 100% answered "no."

1.3.8 Is there anything else I should know about the policies in this course? Responses here included positive feedback about allowing LLMs, such as "I think that these policies create an open learning environment [...] and gives us more practice with AI models."

1.4 Conclusion and Future Work

Incoming students are increasingly familiar with LLMs and prompt engineering, and allowing LLMs is becoming as necessary as allowing students to speak to each other: using LLMs has become a part of learning and writing code. The policy in the authors' current courses is to treat LLMs like we do humans: on assignments where collaboration with humans is allowed, students may talk to an LLM like a human. Just as students may not copy code from another person, students may not use code written by an LLM.

ACKNOWLEDGMENTS

My heartfelt gratitude to my students who let us experiment on them. Thank you also to Mark Miller and Alvaro Monge.

REFERENCES

[1] Joseph Beck, Mia Stern, and Erik Haugsjaa. 1996. Applications of AI in Education. *XRDS: Crossroads, The ACM Magazine for Students* (1996).
[2] Lijia Chen, Pingping Chen, and Zhijian Lin. 2020. Artificial Intelligence in Education: A Review. *IEEE Access* 8 (2020), 75264–75278. https://doi.org/10.1109/ACCESS.2020.2988510
[3] Ira P. Goldstein and Mark L. Miller. 1976. AI Based Personal Learning Environments: Directions for Long Term Research. AI Memo 384. *Institution of Education Sciences* (1976).
[4] Alex Guilherme. 2019. AI and education: the importance of teacher and student relations. (2019).
[5] Leo Porter, Cynthia Bailey Lee, and Beth Simon. 2013. Halving Fail Rates Using Peer Instruction: A Study of Four Computer Science Courses. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 177–182. https://doi.org/10.1145/2445196.2445250
[6] Cynthia Solomon, Brian Harvey, Ken Kahn, Henry Lieberman, Mark L. Miller, Margaret Minsky, Artemis Papert, and Brian Silverman. 2020. History of Logo. *Proc. ACM Program. Lang.* 4, HOPL, Article 79 (jun 2020), 66 pages. https://doi.org/10.1145/3386329
[7] Fati Tahiru. 2021. AI in Education: A Systematic Literature Review. (2021).

Effects of Prior Academic Experience in Introductory Level Data Science Course

Noshaba Bhalli
nbhalli1@umbc.edu
University of Maryland Baltimore County
MD, USA

Vandana Janeja *
vjaneja@umbc.edu
University of Maryland Baltimore County
MD, USA

David Harding
harding@berkeley.edu
University of California, Berkeley
CA, USA

ABSTRACT

Data Science is an in-demand skill in the job market. To meet the demand, universities are offering data science courses or programs. These courses/programs not only give students data science skills, but also awareness of the field, increasing the likelihood of their opting for data science as a career. In this work we look at how the students' prior academic experience affects the outcome of an introductory data science class that requires no pre-requisites. We conducted the study in an undergraduate introductory data science class (IS 296) at the University of Maryland, Baltimore County (UMBC). The course was adapted from University of California Berkeley's Data 8 course. A pre and post survey was conducted to measure four factors in light of Social Cognitive Career Theory (SCCT): self-efficacy, identity, motivation and belonging uncertainty of students as a data scientist before and after taking the class. Our results show that although the course was designed requiring no pre-requisites, students with no prior programming experience and no statistics experience showed decrease in the four factors as compared to students with some prior experience.

ACM Reference Format:
Noshaba Bhalli, Vandana Janeja, and David Harding . 2024. Effects of Prior Academic Experience in Introductory Level Data Science Course . In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635505

1 INTRODUCTION

Increase in data generation, storage, and processing capabilities has increased the demand for data science related jobs over the past decade. US Bureau of Labor Statistics, predicted 36% growth in data science related jobs between 2021 and 2031 [1]. To meet this demand, universities have started offering introductory data science courses at the undergraduate level to expose students to the field. As these courses are new and in some cases adapted from other universities, there is a need to see the effectiveness of these courses across students from different backgrounds so that potential improvements can be made to create equal opportunities for all.

*Corresponding Author

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635505

In this paper we report the effectiveness of an introductory level data science course, IS 296 at the University of Maryland, Baltimore County (UMBC), adapted from UC Berkeley's Data 8 course requiring no pre-requisites. We consider prior academic experience in terms of: exposure to programming, exposure to statistics, and being a transfer student in the university. To measure the effectiveness of the offered course we look at **four factors** in light of social cognitive theory (SCCT): Self efficacy in data science, identity as a data scientist, belonging uncertainty in the data science setting, and motivation for data science (defined in section 2).

Motivation and Research Question The fours factors of SCCT are indicative of individuals' beliefs about their ability to pursue and succeed in a career path. Looking at these factors can help us understand performance of students from different backgrounds and introducing interventions for the under performing groups in terms of the SCCT factors, such as making appropriate changes in curriculum or introducing a module for foundational elements at the beginning of the course. We investigate:

RQ: "*What are the effects of prior academic preparation on the four factors (self-efficacy, identity, motivation and belonging uncertainty) in an introductory level data science course with no pre-requisites?*"

2 METHODOLOGY

A pre-survey was given to IS 296 students at the start of semester and then the same survey was given at the end of the semester (post survey) to measure the change in self-efficacy, identity, motivation and belonging uncertainty. The survey anonymously recorded the demographics of students, to measure differences amongst students from different backgrounds.

Self-efficacy is individual's belief in their own ability to do something, measured by the following seven questions: i) I believe I will receive an excellent grade in this class. ii) I'm certain I can understand the most difficult material presented in the readings for this course. iii) I'm confident I can learn the basic concepts taught in this course. iv) I'm confident I can understand the most complex material presented by the instructor in this course. v) I'm confident I can do an excellent job on the assignments and tests in this course. vi) I'm certain I can master the skills being taught in this class. vii) I'm confident I can find the information I need to overcome programming challenges [2].

Data science identity describes how students think data science is related to who they are, measured by the following three questions: i) I see myself as a data science person. ii) I could imagine myself doing data science in the future. iii) In general, being good at data science is an important part of my self-image [3].

Belonging uncertainty measures the feeling of not belonging in

	Pre		Post	
Total responses	17		14	
	No	**Yes**	**No**	**Yes**
Prior programing experience	8	11	8	6
Prior statistics experience	8	11	6	8
Transfer student	7	10	5	9

Table 1: Distribution of students in the pre and post surveys

a setting, measured by following three questions: i) Sometimes I feel that I belong in data science, and sometimes I feel that I don't belong. ii) When something bad happens, I feel that maybe I don't belong in data science. iii) When something good happens, I feel that I really belong in data science [4]. The response option for all questions above was on scale of 1-7, 7 being strongly agree and 1 being strongly disagree. Belonging uncertainty questions were reverse scored to keep consistency in scoring of all four factors.

Students' Motivation describes students' interest to achieve a goal, measured by questions: For each of the following statements, please indicate how important it was in your decision to take a data science course: i) Data science meets a requirement for my major. ii) Data science skills are important to my major. iii) Data science skills will improve my chances of getting a good job after graduation. iv) Data science skills are important to my extra-curricular activities. v) Data science skills will help me make an impact and solve problems in society. vi). Data science is intellectually interesting to me. vii). Data literacy is important for everyone these days no matter what their career or major. The seven questions were on scale of 1-5, 1 being *Not at all important* and 5 being *Extremely important*.

3 RESULTS

We report the results of a pilot study done in Fall 2022 with 23 registered students (see Table 1 for survey responses by groups).

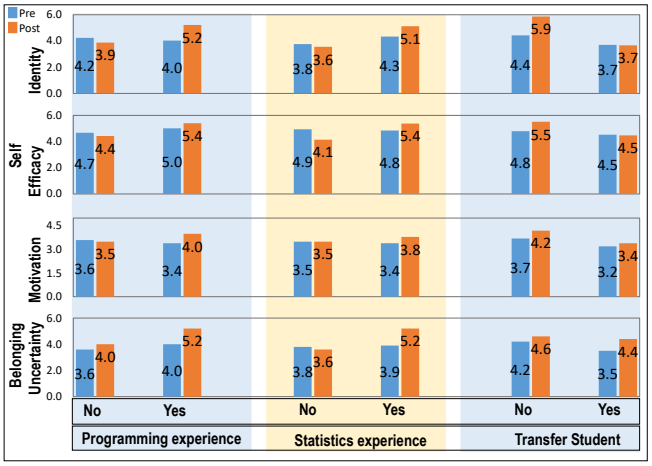


Figure 1: Pre and post survey comparison

We compared the pre and post survey responses to measure identity, self efficacy, motivation, and belonging uncertainty score with respect to prior experience in programming, prior experience in statistics, and transfer student status as shown in Figure 1.

Prior Programming Experience: Students with some prior programming experience showed improvement in identity, self efficacy, motivation, and belonging uncertainty (by 30%, 8%, 18%, and 30% resp.). In contrast, students with no prior exposure to programming showed decreases in identity, self efficacy, and motivation (by -9%, -5%, and -3% resp.) . Only belonging uncertainty improved in these students (by 11%). This decrease in majority of measures shows that programming experience is potentially necessary for the class.

Prior Experience in Statistics: Students with no exposure to statistics showed decreases in identity, self efficacy and belonging uncertainty (by -5%, -16%, and -5% resp.), the motivation measured the same. Where as students with prior experience in statistics showed improvement in all four measures. This makes prior knowledge in statistics potentially important for the students' interest. [5] reports similar findings based on a focus group study where IS 296 students without prior programming experience reported difficulty in programming assignments. All students in our survey with prior experience had it from taking course either in high school, community college or university and/or through self learning.

Transfer student: Previous work shows transfer students' performance decreasing in a new setting [6] as these students have limited/no prior experience in the new institution. We saw mixed trends for transfer vs non-transfer students. There is no negative change for the transfer students taking this class, however, all four factors are higher for non-transfer students than transfer students. Transfer students' identity and self-efficacy scored remained same, belonging uncertainty motivation showed positive improvement.

4 CONCLUSION AND FUTURE WORK

We outlined the effects of students' academic backgrounds on their motivation, identity, self efficacy, and belonging uncertainty in data science settings after taking an introductory data science class. Our results show that no prior experience in programming and/or statistics negatively impacts these four measures. Including introductory content at the beginning of the course, such as a week-long programming and statistics module, can potentially help students. Future work will look at data from additional semesters to develop a robust course plan supporting students from different backgrounds.

ACKNOWLEDGMENTS

This work is supported by NSF grant # 1915714

REFERENCES

[1] U.S bureau of labor statistics (2022) url:. https://www.bls.gov/ooh/math/data-scientists.htm, 2022. Accessed: 2023-09-06.
[2] Paul R Pintrich et al. A manual for the use of the motivated strategies for learning questionnaire (MSLQ). 1991.
[3] Gloriana Trujillo and Kimberly D Tanner. Considering the role of affect in learning: Monitoring students' self-efficacy, sense of belonging, and science identity. *CBE—Life Sciences Education*, 13(1):6–15, 2014.
[4] Gregory M Walton and Geoffrey L Cohen. A question of belonging: race, social fit, and achievement. *Journal of personality and social psychology*, 92(1):82, 2007.
[5] Vandana Janeja, Maria Sanchez, Yi Xuan Khoo, Vacano Claudia, and Karen Chen. Adopting foundational data science curriculum with diverse institutional contexts. In *Proc. of the 55th ACM Technical Symposium on Computer Science Education*, 2024.
[6] Philip E Carlan and Ferris R Byxbe. Community colleges under the microscope: An analysis of performance predictors for native and transfer students. *Community College Review*, 28(2):27–42, 2000.

Do Hints Enhance Learning in Programming Exercises? Exploring Students' Problem-Solving and Interactions

Giang Bui
University of Toronto Mississauga
Mississauga, Canada
giang.bui@utoronto.ca

Nicholas Susanto
University of Toronto Mississauga
Mississauga, Canada
nicholas.susanto@mail.utoronto.ca

Naaz Sibia
University of Toronto
Toronto, Canada
naaz.sibia@utoronto.ca

Angela Zavaleta Bernuy
University of Toronto
Toronto, Canada
angelazb@cs.toronto.edu

Michael Liut
University of Toronto Mississauga
Mississauga, Canada
michael.liut@utoronto.ca

Andrew Petersen
University of Toronto Mississauga
Mississauga, Canada
andrew.petersen@utoronto.ca

ABSTRACT

Asking for help (help-seeking) is a recognized and effective problem-solving strategy. This study investigates students' interaction with on-demand hints (automated hints requested by students) and assesses their impact on learning progress. We conducted an A/B experiment in a third-year computer science database course, offering hints for selected SQL problems with different hint designs. We collected data on students' code submissions, grades, and hint requests, and we administered a survey to gather feedback and gauge student perception of the hints. Many students accessed hints immediately without attempting the problem first, often requesting multiple hints in quick succession. While students perceived the hints to be valuable, we did not detect an impact on student problem-solving. These insights could inform future studies on the possible impact of students' attitudes toward hints, and how different types of hints might impact uptake and perception of hints.

ACM Reference Format:
Giang Bui, Nicholas Susanto, Naaz Sibia, Angela Zavaleta Bernuy, Michael Liut, and Andrew Petersen. 2024. Do Hints Enhance Learning in Programming Exercises? Exploring Students' Problem-Solving and Interactions. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635563

1 INTRODUCTION

Automated hints are valuable resources for students in programming courses, with two primary delivery methods: on-demand hints, initiated by students themselves [1], and unsolicited hints, provided to students when the system believes them valuable [2]. Recognizing the significance of hints in supporting students' learning journeys, we integrated on-demand, automated hint options into our online programming exercise system. Our research aims to further investigate the role of hints in enhancing students' success and their patterns of hint utilization. Specifically, we investigate the following two research questions:

RQ1: How do students perceive and utilize hints when solving programming problems?

RQ2: How does the presence of hints relate to the number of code submissions students make to obtain a correct answer?

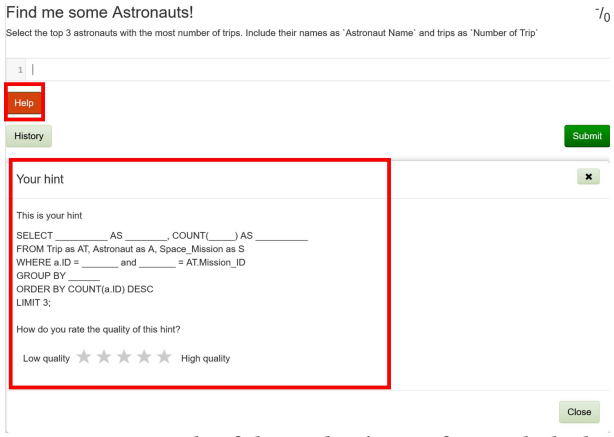


Figure 1: An example of the student's interface with the hint button and content highlighted

2 METHODS

In a third-year computer science databases course, we offered hints to students for three SQL query writing exercises [3]. To assess their impact, we conducted a controlled study where students were randomly assigned into two groups (75% of students saw the button to request hints, while the remaining 25% did not receive any hints). Those conditions were assigned randomly and independently (two-step randomization). Hints provided were either text suggestions (text hints) or skeleton code (code hints, similar to the hint in Figure 1). The hints were written by instructors and teaching assistants in the course.

All data was de-identified. 190 students submitted a total of 5,826 responses for three problems, generating 1,445 hint requests. Data from our hint server included experiment conditions, student feedback, and hint request timestamps. Our online exercise system recorded the students' code submissions, timestamps, and scores.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635563

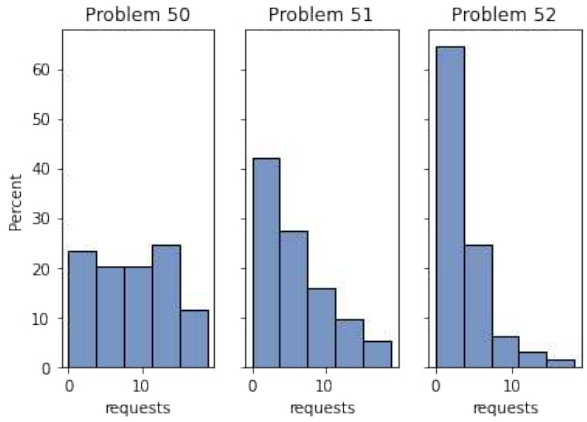


Figure 2: Number of total student code submissions before requesting the first hint for problems 50, 51, and 52

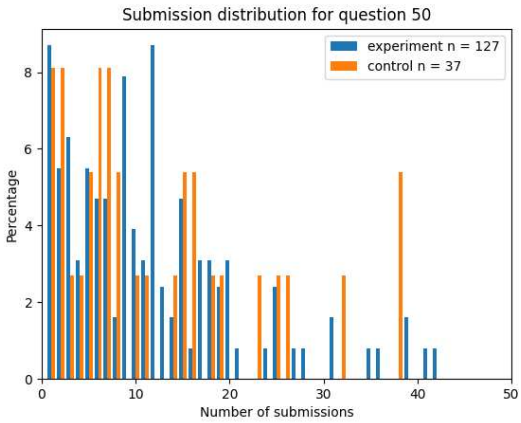


Figure 3: Number of total submissions until a perfect score for problem 50

A voluntary end-of-term survey with 183 responses assessed student perceptions and feedback on hints. We used a 5-item scale (range from "Strongly Disagree" to "Strongly Agree") to measure the students' agreement with statements (e.g., "The hint resource provided effective tips about solving the exercise.").

3 RESULTS

3.1 Students' use and perception of hints

Out of 186 students in the experiment group (with access to hints), the majority requested hints between 1 and 8 times across the three problems. This trend held true for each problem.

Our analysis indicates that students generally sought hints early when hints were available. We observed evidence of students who requested hints sooner once they became aware of their availability (Figure 2) – students had fewer attempts before asking for a hint in later problems than in the first problem with hints ($F(2, 254) = 11.41$, $\boldsymbol{p < .001}$, $\eta^2 = 0.082$). We also found that most students requested a second hint within 10 seconds of receiving a hint. Given the hint lengths, this raises concerns about the adequacy of the time students spend on reading and reflecting upon each hint. This behavior could suggest help abuse or a desire for the hint to be a direct solution.

66.24% (104 survey respondents) who were aware of and may have used the hints found them helpful. More than half of them agreed that hints improved their understanding (56.69%) and problem-solving ability (63.06%). Of the 65 students who provided feedback on hint quality using a 5-item scale, the majority (70.77%) rated the hints as high-quality (above 3 out of 5). Notably, text hints received lower feedback scores compared to code hints ($N_{code} = 44$, $\mu_{code} = 4.52$; $N_{text} = 21$, $\mu_{text} = 3.62$; $U = 599.0$, $\boldsymbol{p = 0.017}$).

3.2 Hints and students' problem-solving

We did not observe noticeable differences in the number of attempts between students in the control and experiment groups. For question 50, the experiment group required fewer attempts on average to achieve a perfect score ($\mu = 13.3$, $N = 127$) compared to the control group ($\mu = 17.2$, $N = 37$) (Figure 3). However, a Wilcoxon rank-sum test provided no evidence of a difference ($p = .27$, $r = 0.28$). Similar results were observed for questions 51 and 52.

A post-hoc power analysis revealed that our study lacked statistical power to identify results with small effect sizes. For example, questions 50 and 52 had estimated effect sizes of 0.28 and 0.15, respectively, suggesting that the sample sizes for both questions (N=164 & N=169) may have been insufficient to detect true statistical differences. With a significance level of α=0.05 and 80% power, a minimum sample size of 207 and 670 participants would have been required. However, we have sufficient participants to demonstrate a lack of moderate-size effects.

4 CONCLUSION AND FUTURE WORK

Our study investigates students' usage of automated, on-demand hints (hints requested by students) and assesses the impact of hints on student learning progress. Most students accessed hints immediately without attempting the problem first, especially when they got used to the hint options. Students also often requested multiple hints in quick succession. Although students perceived the hints to be valuable for their learning and to be of high quality, we did not detect an impact on students' efficiency in approaching the solution. Our future work will investigate whether different types of hints have a different impact and investigate other learning gains from the hints such as students' progress and problem-solving.

REFERENCES

[1] Paolo Antonucci, Christian Estler, Durica Nikolić, Marco Piccioni, and Bertrand Meyer. 2015. An Incremental Hint System For Automated Programming Assignments. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (Vilnius, Lithuania) *(ITiCSE '15)*. Association for Computing Machinery, New York, NY, USA, 320–325. https://doi.org/10.1145/2729094.2742607

[2] Mehak Maniktala, Christa Cody, Tiffany Barnes, and Min Chi. 2020. Avoiding Help Avoidance: Using Interface Design Changes to Promote Unsolicited Hint Usage in an Intelligent Tutor. *International Journal of Artificial Intelligence in Education* 30, 4 (01 Nov 2020), 637–667. https://doi.org/10.1007/s40593-020-00213-3

[3] Daniel Zingaro, Yuliya Cherenkova, Olessia Karpova, and Andrew Petersen. 2013. Facilitating Code-Writing in PI Classes. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 585–590. https://doi.org/10.1145/2445196.2445369

A Comparative Analysis of Tools & Task Types for Measuring Computational Problem-Solving

Engin Bumbacher
engin.bumbacher@hepl.ch
Haute École Pédagogique du Canton de Vaud (HEP-VD)
Lausanne, Vaud, Switzerland

Jérôme Brender
jerome.brender@hepl.ch
Haute École Pédagogique du Canton de Vaud (HEP-VD)
Lausanne, Vaud, Switzerland

Richard Lee Davis
richard.davis@epfl.ch
École Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Vaud, Switzerland

ABSTRACT

How to measure students' Computational Problem-Solving (CPS) competencies is an ongoing research topic. Prevalent approaches vary by measurement tools (e.g., interactive programming, multiple-choice tests, or programming-independent tests) and task types (e.g., debugging problems or Parson problems). However, few studies have examined the measurement tools of CPS competencies themselves: affordances and limitations of the measurement tools and how they compare, or whether different task types might elicit CPS competencies differently. Research needs to address these questions in order to better understand how to design robust, generalizable, and effective measurement tools for CPS competencies. This paper presents an exploratory study that contributes to this research direction. It is part of a larger international project to develop an open-access formative assessment platform for CPS, which includes a novel authoring tool for a wide range of task types for interactive block-based programming. We used the tool to create an interactive programming experience with multiple task types and gave it to more than 300 secondary school students from different countries. We also administered a validated multiple-choice measurement of Computational Thinking with block-based programs. We focused on task complexity as a characteristic of task type, using a classification scheme based on task design features. Comparing students' performances on tasks of different complexity and using two distinct measurement tools, we found that the multiple-choice measurement only partially predicts performance in the interactive programming task. Additionally, its predictive capacity varies significantly between task types of differing complexity.

ACM Reference Format:
Engin Bumbacher, Jérôme Brender, and Richard Lee Davis. 2024. A Comparative Analysis of Tools & Task Types for Measuring Computational Problem-Solving. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635547

1 FURTHER INFORMATION

Computational Thinking (CT) is widely recognized as a crucial skill in the twenty-first century [2]. It is involved in the formulation, evaluation, and implementation of solutions to complex computational problems [3, 8]. There is an increasing number of learning environments and pedagogical approaches for teaching computational problem-solving (CPS), but also a lack of agreed-upon tools for measuring CT practices in CPS - tools that are necessary to help advance research [4]. Current approaches range from multiple-choice tests using programming-based tasks to interactive experiences with novel immersive virtual environments. In one of the more comprehensive studies of the few that examined the relationship between different measurement tools, Román-González et al. [6] found only a partial convergence between the most extensively used tools: the psychometrically validated paper-based assessment CTt, the Dr. Scratch [5] scoring tool for Scratch programs, and the programming-independent paper-based assessment of CT, Bebras [1]. This indicates that each test captures only some aspects of CT practices, leaving unexplained a significant proportion of other ones. A combination of the different measurement tools, as suggested by the authors, will not resolve this problem as long as we do not better understand what aspects of CT practices each measurement tool can and cannot capture, and how they relate to each other. In other words, we need to be able to answer questions such as whether a debugging task in multiple-choice format, as used in the CTt, captures the same aspect of debugging practices as an interactive programming task, or whether performance on a CPS task varies when changing design features of the task such as the degrees of freedom in finding a solution, the kind of interactivity in the task, etc.

This paper presents an exploratory empirical comparison of different measurement approaches to begin to address these questions, in order to contribute to more informed methodologies for designing future measurement approaches. Specifically, we address the following research question: To what extent does performance on a validated multiple-choice instrument of CT predict performance on interactive CPS tasks, and does this relationship depend on the complexity of the programming task?

In order to address this question, we used a novel web-based Platform for Innovative Learning Assessments (PILA) in collaboration with the OECD's Programme for International Student Assessment (PISA). This platform allows the creation and distribution of highly customized assessment experiences with interactive CPS tasks and records all process data. We worked with the "Karel" programming application, a block-based programming environment that requires students to guide the turtle-like agent "Karel" through a grid-like world while arranging stones in certain ways. The novel authoring tool allows for varying the complexity of the tasks by manipulating a wide range of design features such as the number, size and shape of the grid-like worlds to be solved with a program, the types of

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635547

programming blocks available, the limits to block use, the available starter code, etc.

We designed two assessment experiences containing eight and six interactive CPS tasks of varying complexity. Tasks required students complete, assemble, debug or modify given code, involved simple or more complicated worlds. Throughout the experiences, three programming concepts were introduced: functions, while loops, and conditionals. Using an a priori coding scheme for task complexity, which we will present in the poster, we classified each task as being of "low complexity" or of "high complexity". Task complexity was varied within each assessment experience.

The participants were 305 secondary school students between 12 and 18 years from a total of 28 teachers in various schools from five countries (Singapore, Ireland, the Netherlands, and Brazil). These schools responded to an open call for participation by the OECD's PISA. Prior to the start of the study, ethical approval was secured, but we could not collect any background information on the students.

The study was structured in three parts. Initially, students completed a self-report questionnaire on their prior computer science and programming experience. Subsequently, they completed the B+CTt paper-based multiple-choice test developed by Wiebe et al. [7]. After a brief tutorial on using the "Karel" application, students worked through the two 45-minute assessment sessions on consecutive days.

For each student, we calculated a B+CTt test score as the percentage of the maximum score possible, a compound measure of prior expertise as the aggregate of students' responses to three closed-form self-report questions on prior programming experiences, and a programming score for the performance in the CPS assessment experiences. Both the success rate and the percentage of students who could attempt a task varied significantly between tasks, with an average overall score of 45% (SD=22%) across both experiences. The rate of non-attempts increased with each subsequent task within an experience. At the same time, the 7 low complexity and 7 high complexity tasks were arranged differently within each assessment experience, with the low complexity tasks being placed mostly in the first half of an experience. Thus, the average rate of non-attempts was significantly different between low and high complexity tasks. This led us to decide to run separate regression models for the two categories instead of combining them into a single model with the task category as an independent interactive factor. Qualitative comparisons of coefficients between models are possible because we used standardized covariates. We used linear mixed-effect models on performance in programming tasks, with random intercepts for country and for teachers nested in country. Table 1 shows the three mixed-effects models, with dependent variables total score on all tasks, total score on just the low complexity tasks, and total score on just the high complexity tasks. The results show that both the B+CTt score and the compound measure of prior expertise are significantly predictive of performance on the interactive programming tasks in all three models. However, while the main effect of prior expertise remains fairly similar across models, the main effect of B+CTt score varies significantly, and is highest for the low complexity tasks. Similarly, the interaction effect between the two covariates varies across models, and is highest for the high complexity tasks. These results suggest there is partial convergence between the B+CTt and the interactive programming experiences. However, the strength of prediction of the B+CTt for performance on the Karel tasks varies with task complexity. There is a larger proportion of variance in performance that cannot be explained by the B+CTt for tasks of high complexity compared to tasks of low complexity. Taken together, this exploratory study emphasizes the importance of paying more attention to how task characteristics such as complexity influence how students engage in computational problem-solving practices and how they perform across different measurement tools. Further research is needed to better understand the relationship between task complexity and predictive capacity of the paper-based closed-response tasks. This requires research tools like the novel web-based PILA, that we are co-developing with the OECD's PISA organization. We will present more details in the poster on both the web-based platform that we used and the study design and results.

Table 1: Mixed-effects models for total scores per task category. Covariates have been standardized. Numbers in parentheses indicate the 95% confidence intervals.

	All Tasks	**Low Complexity**	**High Complexity**
	Total Score	*Total Score*	*Total Score*
(Intercept)	0.09 (-0.32–0.51)	0.00 (-0.12–0.12)	0.12 (-0.40–0.63)
B+CTt Score	0.51 *** (0.40–0.63)	0.60 *** (0.50–0.71)	0.40 *** (0.28–0.51)
Prior Expertise	0.14 *** (0.07–0.21)	0.14 ** (0.05–0.23)	0.12 ** (0.05–0.19)
B+CTt Score:Prior Exp.	0.07 * (0.00–0.13)	-0.06 (-0.14–0.02)	0.16 *** (0.10–0.23)
Random Effects			
σ^2	0.32	0.53	0.33
ICC	0.37	0.08	0.46
Marginal / Cond. R^2	0.38 / 0.61	0.42 / 0.46	0.27 / 0.60

*$p<0.05$ **$p<0.01$ ***$p<0.001$

REFERENCES

[1] Valentina Dagiene and Gabriele Stupuriene. 2016. Bebras–A Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in education* 15, 1 (2016), 25–44.

[2] Shuchi Grover and Roy Pea. 2013. Computational thinking in K–12: A review of the state of the field. *Educational researcher* 42, 1 (2013), 38–43.

[3] Shuchi Grover and Roy Pea. 2018. Computational thinking: A competency whose time has come. *Computer science education: Perspectives on teaching and learning in school* 19, 1 (2018), 19–38.

[4] Josef Guggemos, Sabine Seufert, and Marcos Román-González. 2023. Computational thinking assessment–towards more vivid interpretations. *Technology, Knowledge and learning* 28, 2 (2023), 539–568.

[5] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2015. Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia* 46 (2015), 1–23.

[6] Marcos Román-González, Jesús Moreno-León, and Gregorio Robles. 2019. Combining assessment tools for a comprehensive evaluation of computational thinking interventions. *Computational thinking education* (2019), 79–98.

[7] Eric Wiebe, Jennifer London, Osman Aksit, Bradford W Mott, Kristy Elizabeth Boyer, and James C Lester. 2019. Development of a lean computational thinking abilities assessment for middle grades students. In *Proceedings of the 50th ACM technical symposium on computer science education*. 456–461.

[8] Jeannette Wing. 2017. Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology* 25, 2 (2017), 7–14.

Integrating & Implementing K-12 Computing Pathways across Six School Districts—Challenges & Opportunities

Quinn Burke
Learning Experience Design
Digital Promise Global
Redwood City CA
qburke@digitalpromise.org

Merijke Coenraad
Learning Experience Design
Digital Promise Global
Washington DC
mcoenraad@digitalpromise.org

Alessandra Rangel
Learning Experience Design
Digital Promise Global
Redwood City CA
arangel@digitalpromise.org

Kyle Dunbar
Learning Experience Design
Digital Promise Global
Washington DC
kdunbar@digitalpromise.org

Kelly Mills
Learning Experience Design
Digital Promise Global
Washington DC
kmills@digitalpromise.org

ABSTRACT

This SIGCSE poster presents a landscape study of six school districts across six different states in terms of their individual teachers' and administrators' capacity to integrate and implement computational thinking (CT) into their own schools and classrooms. This landscape evaluation represents the baseline start of a wider, four-year national study around district capacity to collaboratively develop consistent and comprehensive K-12 computing pathways for their students and schools. The early landscape work we present here not only represents a starting point for comparing district educators' comprehension of CT (and computer science [CS]), but also acts as an early indicator as to what extent K-12 computing is a school and district-based priority, and to what degree teachers feel they have the capacity to meaningfully implement it. This poster relies on two data sources in a mixed-methods design: Districtwide surveys of teachers and administrators on their familiarity and prioritization of CT, coupled with subsequent hour-long focus group discussions with educators to expand upon their respective district landscape survey responses. Results point to all districts perceiving the broad applicability of CT as a skill set and its integrative potential in a range of subjects. Yet in terms of classroom implementation, teachers find such CT integration decidedly less clear, recognizing it to be a priority but also reporting as less confident about creating their own curricular materials, where CT situates with their district's ongoing initiatives, and where they can find curricular resources and tools specific to their own areas of curricular integration.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03. https://doi.org/10.1145/3478432.3499076

ACM Reference format:
Quinn Burke, Merijke Coenraad, Alessandra Rangel, Kyle Dunbar, & Kelly Mills. 2024. Integrating & Implementing K-12 Computing Pathways across Six School Districts—Challenges & Opportunities. In *Proceedings of 2024 Special Interest Group on Computer Science Education (SIGCSE'24), March 20-23, Portland, OR.* ACM, NY, NY, USA, 2 pages. https://doi.org/10.1145/3478432.3499076

1 INTRODUCTION / OVERVIEW

To date, a growing number of school districts are eager for computer science and computational thinking offerings. A recent 2021 study conducted by Gallup and Google found that 58% of district superintendents agree that CS is a top priority in their district, up from 33% just four years ago [2]. Yet this same Gallup/ Google report also points to the lingering and stubborn disparities in participation: Only 46% of Black students and 46% of Hispanic students report access to computing coursework, compared to 52% percent of white students. Research [1; 5] specifically points out that these disparities do not commence in high school but start as early as lower elementary school, in which some districts/schools offer robust computing offerings while others persist using devices solely for "skill-and-drill" routines. Research demonstrates that integrating CT on the elementary and middle school levels is not only possible but successful in a range of subject areas, including science and mathematics [8], language arts [7], social studies [6], and music [3]. The integrative potential of CT however also has created problems for schools, as the the persistent lack of clarity around what exactly constitutes CT has left districts stubbornly bereft of a meaningful definition. And the *integration of CT* on the elementary and middle school levels also presents challenges in terms of the actual *implementation of CT.*

The purpose of this poster is to further (and more systematically) pursue this tension between integration and implementation when it comes to CT offerings This poster addresses two central questions. *First, in terms of integrating CT into their classrooms, are teachers still finding this to be a priority, and, if so, are they also identifying viable integration points in the curricula? Second, in terms*

of implementation, what challenges are associated with offering CT in a range of subjects and what do teachers need in terms of support?

2 METHODS

2.1 Participating School Districts

The table below provides an overview of the districts in this study:

District Name	Size/ Schools	District Demographics	Participants
Broward County Public Schools FL	Large Urban 329 schools 270,000 students	Title I: 58% Black: 39% Latinx: 35%	252 teachers (surveys) 7 teachers (focus groups)
Indian Prairie School District IL	Suburban 31 schools 28,000 students	Title I: 31% Black: 3% Latinx: 31%	65 teachers (surveys) 7 teachers (focus groups)
Iowa City School District IA	Suburban 28 schools 14,500 students	Title I: 65% Black: 23% Latinx: 13%	59 teachers (surveys) 7 teachers (focus groups)
Kettle Moraine Public Schools WI	Suburban 11 schools 3,900 students	Title I: 10% Black: 1% Latinx: 5%	59 teachers (surveys) 7 teachers (focus groups)
North Salem School District NY	Suburban 2 schools 1068 students	Title I: 12% Black: 2% Latinx: 15%	112 teachers (surveys) 5 teachers (focus groups)
Talladega County Schools AL	Rural 17 schools 7,000 students	Title I: 57% Black: 3% Latinx: 2%	14 teachers (surveys) 6 teachers (focus groups)

Table #1: District Overviews

2.2 Data Collection & Analysis

Surveys were administered via email among districts in March and April of 2023 and included a range of teachers in terms of grade level (elementary, middle, and high school), experience offering CS/ CT, as well as ensuring teachers and administrators from all schools in the district were included. The follow-up focus groups occurred over later Spring and Summer(April – September 2023). Focus groups for each district took between 45-70 minutes and occurred via Zoom. They likewise included a range of teachers, though these participants were specifically identified by district leaders as having experience teaching/ supporting CS/ CT.

3 RESULTS

Given the limited space for SIGCSE poster proposals, we cannot fully expound on each district's current landscape offerings, and educators' sense of capacity to implement their district's wider K-12 trajectory. This stated, results will be structured around our two original research questions (above) around district's **integration** and district's actual **implementation** of CT, respectively: In terms of district **integration** of K-12 computing, all six districts are clearly integrating CT in a range of subjects. To this point, the poster will display the range of subject matters in which each district is integrating CT via a series of multi-colored pie charts, highlighting what subjects are most prevalent points of integration and which are less common. In terms of leading subjects in which CT/ CS is currently being integrated across the partnering six districts, science and math are most prominent on the middle and high school level, while ELA and social studies fall into third and fourth place respectively, and are more characteristic of elementary and lower middle school integration across the six districts. On average, 57% of teacher respondents across the six districts said they integrate into their disciplinary teaching, 72% of whom indicated such integration occurred within specific tasks and 54% of whom said it happened within open-ended creative tasks. Teachers from all six districts likewise indicated that such subject matter integration was a district-wide priority with a mean response of 78% of them "agreeing" or "strongly agreeing" it was crucial learning for all their students. In terms of district **implementation**, preliminary results show a more complicated picture among the six districts. While teachers recognize CT as a key skill and CS as an important academic discipline, they are less confident (<50%) computing remains a broader districtwide priority, and they report being significantly less confident in developing their own lessons in CT (only 13% reporting strong confidence) than in their general understanding of CT as a whole (26%). Moreover, while teachers recognized the need to broaden participation (74% mean), they reported to be significantly less equipped (<10% across districts) to design and implement culturally relevant computing lessons and activities.

4 CONTRIBUTIONS & FUTURE WORK

This work expands research on district-level computing pathways especially focused on promoting student inclusivity and growing more consistent and cumulative pathways to meaningful participation. Our focus on district capacity is also significant, for though all 50 states have enacted K-12 CS policies to date [4], to what degree districts are taking ownership of CS standards and graduation requirements remains, at best, a mixed picture.

REFERENCES

[1] Gallup Inc. & Google Inc. 2021. Current perspectives and continuing challenges in CS education in U.S. K-12 Schools. https://csedu.gallup.com/home.aspx.

[2] Kelly Mills, Merijke Coenraad, Pati Ruiz, Quinn Burke, and Josh Wesigrau. (2021). Computational thinking for an inclusive world: a resource for educators to learn and lead. *Digital Promise*. DOI: 10.51388/20.500.12265/138.

[3] Christopher Petrie. 2022. Programming music with Sonic Pi promotes positive attitudes for beginners. *Computers & Education, 179*, 104-09.

[4] Cassandra Scharber, Lana Peterson, Yu-Hui Chang, Sarah Barksdale, and Ramya Sivaraj. 2021. Critical computing literacy: Possibilities in K-12 computer science education. *Pedagogies, 16*, 2, 136-151.

[5] Mia Shaw & Yasmin B. Kafai, Y.B. 2020. Charting the identity turn in K-12 computer science education. In *14th International Conference of the Learning Sciences (ICLS) 2020, Volume 1*, 114-121. Nashville, TN.

[6] Tamara L. Shreiner and Mark Guzdial. 2022. The information won't just sink in: Helping teachers provide technology-assisted data literacy instruction in social studies. *British Journal of Educational Technology 53*, 5, 1134–1158.

[7] Sara Vogel, Christopher Hoadley, Laura Ascenzi-Moreno, and Kate Menken. 2019. The role of translanguaging in computational literacies. *SIGCSE 50th ACM Technical Symposium*, ACM Press, New York, NY, 1164–1170.

[8] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2016. Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25,1, 127–147. DOI: https://doi.org/10.1007/s10956-015-9581-5.

Analyzing Student and Instructor Comments using NLP

Zack Butler
zjb@cs.rit.edu
Rochester Inst. of Tech.
Rochester, NY, USA

Ivona Bezáková
ib@cs.rit.edu
Rochester Inst. of Tech.
Rochester, NY, USA

Shaoxuan Xu
cx2336@rit.edu
Rochester Inst. of Tech.
Rochester, NY, USA

Angelina Brilliantova
lb9849@rit.edu
Rochester Inst. of Tech.
Rochester, NY, USA

ABSTRACT

We report on our experience using common natural language processing (NLP) tools to analyze two vastly different data sets of free-form responses collected during a study of assignments in introductory computing courses. Our first data set consists of typically short comments left by hundreds of students on assignment surveys. Our second data set is comprised of semi-structured individual interviews of eight instructors of up to an hour long each. We collected the data across several years as part of our investigation of the use of pencil puzzles as a context for introductory computer science. In an earlier work, we manually analyzed a fraction of the student comments (all data collected until that point), using grounded theory. The results were illuminating, but the process was very time consuming, consisting of manual assignment of a small number of codes to each comment. In this work, we investigate the usability of common NLP tools to speed up the process for the entire data set of student comments. We also applied these tools to the instructor interviews. The NLP tools do not appear to be effective to create the code base, but, once the code base was determined, they performed the actual coding (assignment of codes to each student comment) promisingly well. For the long-form instructor interviews, the situation was much more challenging, due to the wide-ranging nature of semi-structured interviews, interleaving discussion topics, and elements of natural speech. We report on the lessons learned while automatically analyzing these complex data sets.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

Instructor interviews, Student survey comments, Manual qualitative analysis, Natural language processing

ACM Reference Format:
Zack Butler, Ivona Bezáková, Shaoxuan Xu, and Angelina Brilliantova. 2024. Analyzing Student and Instructor Comments using NLP. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635593

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635593

1 INTRODUCTION AND BACKGROUND

Grounded theory [3] is a well-established technique for qualitative analysis of free-form responses. It consists of the creation of a code base — a small set of strings capturing the most common themes in the data set. Then, a small number of codes are associated with each comment or a part of the free-form text (for example, a paragraph or a sentence in an interview). Having thus coded the free-form responses, standard statistical approaches can be applied to study the occurrences and co-occurrences of individual codes, determining correlations between the themes and sentiments in the responses.

The manual creation of the code base (often done using triangulation, when multiple researchers compare their proposed code bases and arrive at a mutually agreeable compromise), and, even more so, the manual assignment of codes to individual comments, paragraphs, or sentences, is very time consuming. We investigate the use of common NLP tools to help automate the process.

We collected our data set over several years, and across several institutions. We studied the use of *pencil puzzles* (puzzles such as sudoku that are designed to be solved by people, whether with a pencil or an app) as a context for introductory CS assignments. We initially collected data (student surveys including Likert-scale questions and open-ended comment boxes, plus grade data) at our home institution. We analyzed the Likert-scale data, and found that students reacted positively to these assignments, and that their experience was independent of their prior exposure to computing or of their gender [1]. We also performed qualitative analysis using grounded theory on the open-ended comments [2]. Notably, this work required manual coding of over 1000 individual comments.

As a followup to the initial study, we recruited 10 collaborating instructors (at institutions ranging from R1 schools to small liberal-arts schools to an international campus of our own university), who delivered puzzle-based assignments in 14 different course sections and allowed us to survey their students. Throughout this second study, we also conducted semi-structured interviews with the collaborating instructors and some of their teaching assistants, reflecting on how they valued the use of puzzle-based assignments and their adoption into their courses. We used grounded theory to manually code the interviews and the student comments in this

larger data set. Recent advances in large language models may allow some of this coding and analysis to be automated, and since we have the human coding available for comparison, we find ourselves in a strong position to evaluate the efficacy of such approaches.

2 RELATED WORK

Natural language processing (NLP) research has achieved admirable progress in recent years. However, the use of these tools in the context of education research has been limited thus far. A survey from 2021 [7] summarizes studies using NLP to analyze the sentiments of students' feedback. The authors identified almost a hundred publications, though only a fraction uses student feedback in the form of survey responses (other sources of student feedback included blog posts, forums, and feedback provided on online education and research platforms). Almost all of this work examines only whether students' comments are generally positive or negative, but we wish to examine whether similar techniques can distill the multitude of thoughts given at the end of a survey. In addition to sentiment analysis, we found papers discussing automated approaches to obtain various statistical analyses such as identifying important keywords and correlations between them, e.g. [5, 8], but we have not found publications aiming for (semi-)automated and more detailed analysis of student feedback.

3 METHODOLOGY

The various themes expressed in student comments in our initial data set included the puzzle concept, the student's learning experience, the details of the assignments, and the overall course. In each of these areas, there were many comments both positive and negative, and our codes reflected the combination of a theme and the accompanying sentiment. Once the codebook was developed, each comment (from both the initial study and the followup study) was assigned between one and three codes based on its content. For the interviews in the followup study, we used a similar approach, though here the coding is more free-form, as large parts of the interview may be irrelevant to the study topic and remain uncoded, while some codes may apply to a long section of the interview.

To perform supervised learning of comment codes, rather than developing new models, we chose to investigate how well common existing language models would be able to learn the coding. We initially chose the BERT model [4] as a well-known and effective deep-network model for general language understanding tasks, but also looked into less computationally intensive models such as FastText [6]. For the interview data, we also used BERT, but this data required significant pre-processing to turn the natural human conversation into a more regular form. We also used various data augmentation techniques on both data sets to increase the size and breadth of the training data.

Finally, we investigated clustering techniques on the survey comments to see if they could generate the initial set of comment codes. Here again we chose to explore a common technique, Latent Dirichlet Analysis (LDA), which is often used for topic modelling across a document corpus. LDA uses a probabilistic model that describes documents as collections of topics, where each topic is represented as a distribution over all words. The algorithm then uses an inference process to determine both the most likely distribution of topic clusters and the most likely distributions of the words in each cluster.

4 RESULTS

For the comment analysis, since each comment has up to three correct codes (as determined by the manual coding), instead of a binary accuracy measurement per comment, we measured success with precision and recall metrics. BERT was able to achieve a precision of 0.660 and recall of 0.750, and manual investigation showed that many "errors" assigned codes similar to the correct ones. (This often happens also with human coding, when different researchers pick up different nuances of the text. The final coding is then the result of a discussion between the researchers.) On the other hand, Fasttext was able to achieve only 0.53 precision and recall.

For the interviews, BERT performed extremely well at determining which parts of the interviews were relevant and which were not. Including the "uncoded" label, the precision of the model was 0.976 and recall was 0.966. However, if we look only at the portions of the interviews which had codes assigned in the ground truth, the accuracy was lower, though still somewhat reasonable, with a precision of 0.410 and recall of 0.667. As such, it appears that NLP techniques can be a meaningful time saver at least in terms of locating the most relevant portions of these long interviews.

On the other hand, clustering techniques appear unable to generate a meaningful set of codes for these survey comments. After manually examining the comments put together in each cluster, there appeared to be little consistency among them. We also compared the manually-assigned codes within each cluster, and likewise found that there was little correspondence between them. In short, it does not appear that this type of analysis is (yet) suitable for topic discovery amongst collections of short comments.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. DUE-1821459. We thank Kimberly Fluet for her valuable guidance on qualitative analysis techniques.

REFERENCES

[1] Zack Butler, Ivona Bezáková, and Kimberly Fluet. 2017. Pencil Puzzles for Introductory Computer Science: An Experience- and Gender-Neutral Context. In *Proceedings of SIGCSE 2017*. 93–98. https://doi.org/10.1145/3017680.3017765

[2] Zack Butler, Ivona Bezáková, and Kimberly Fluet. 2018. Analyzing Rich Qualitative Data to Study Pencil-Puzzle-Based Assignments in CS1 and CS2. In *Proceedings of ITiSCE 2018*. Association for Computing Machinery, New York, NY, USA, 212–217. https://doi.org/10.1145/3197091.3197109

[3] K. Charmaz. 2014. *Constructing Grounded Theory*. SAGE Publications. https://books.google.com/books?id=y0ooAwAAQBAJ

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[5] Explorance 2016. Analyzing student comments in online course evaluations with Blue Text analytics. https://gatorevals.aa.ufl.edu/media/gatorevalsaaufledu/BTA_whitepaper_Analyzing-Student-Comments.pdf.

[6] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759* (2016).

[7] Zenun Kastrati, Fisnik Dalipi, Ali Shariq Imran, Krenare Pireva Nuci, and Mudasir Ahmad Wani. 2021. Sentiment Analysis of Students' Feedback with NLP and Deep Learning: A Systematic Mapping Study. *Applied Sciences* 11, 9 (2021). https://doi.org/10.3390/app11093986

[8] Anna Koufakou, Justin Gosselin, and Dahai Guo. 2016. Using data mining to extract knowledge from student evaluation comments in undergraduate courses. In *2016 International Joint Conference on Neural Networks (IJCNN)*. 3138–3142. https://doi.org/10.1109/IJCNN.2016.7727599

Parallel Islands: A Parallel Computing Educational Video Game

Melissa Cameron
melissacameron@vt.edu
Virginia Tech
Blacksburg, VA, USA

Margaret Ellis
maellis1@vt.edu
Virginia Tech
Blacksburg, VA, USA

Dimitrios Nikolopoulos
dsn@vt.edu
Virginia Tech
Blacksburg, VA, USA

ABSTRACT

The rise in multiprocessors has led to the incorporation of parallel processing in virtually all segments of industry. Creation of and maintenance for the software to run these systems, as well as for the applications using these systems, requires extensive knowledge of the concepts and skills of parallel and distributed computing (PDC). This will naturally lead to an increase in the demand for software developers familiar with PDC and an increase in the demand for universities to incorporate PDC concepts into all levels of their curricula including introductory courses. Because there are real and perceived difficulties in teaching PDC concepts, particularly early in the CS curriculum there is a need to produce educational materials to assist with this expansion. At the same time CS education is wrestling with the surge in the need for graduates with PDC skills, it is also attempting to overcome a gender imbalance in CS. The necessity to create the materials required for increasing PDC education provides an opportunity to make strides in increasing gender diversity in CS as well. Therefore, Parallel Islands was created as a tool to aid in introducing PDC concepts in introductory CS courses in a manner that appeals to all gender identities. It is a web-based educational video game; along with the associated pre- and post-gameplay introductory, review and discussion sections, that is designed to be used by students autonomously or with instructor oversight.

ACM Reference Format:
Melissa Cameron, Margaret Ellis, and Dimitrios Nikolopoulos. 2024. Parallel Islands: A Parallel Computing Educational Video Game. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635591

1 FURTHER INFORMATION

1.1 PDC Education

Traditionally undergraduate PDC education has been incorporated into upper level (junior and senior level) elective courses and perhaps *Operating Systems* or *Computer Architecture* style courses [14]. There has been and is, however, an effort being made to increase the amount of PDC incorporation as well as expand PDC instruction throughout all levels of CS curricula, freshman to senior [1, 2, 9]. An informal review of the course catalogs of a selection of 40 US universities and colleges found 156 courses offered by the schools which were identified as including PDC content. Of those courses only 5 of them were at an introductory level and 7 of them in non-Systems lower level courses indicating that incorporation of PDC content early in CS curriculum has made little progress.

The slow adoption of PDC content into introductory courses in the CS curricula can be attributed to five main challenges. The first is the perceived difficulty of PDC subject matter [3, 12] for CS students. The second is a shortage of qualified instructors [3, 10]. This shortage is particularly acute at introductory levels where the instructors may not have chosen to take parallel computing courses during their own education. The third is the lack of ready-to-adopt instructional materials [10] and the fourth challenge is the need for hardware such as clusters or distributed networks. The fifth challenge is attempting to add new information to the already full schedule of topics to be covered in CS courses [5, 11].

One method of increasing PDC content inclusion in introductory courses may be the generation of high-quality, easily adoptable supplementary educational materials (or modules) that can be incorporated with minimal changes into current courses. Parallel Islands was designed and implemented to be such a module and it addresses many of the challenges leading to slow adoption of PDC content into introductory CS courses as shown in Table 1. It is a multi-level, educational video game that introduces new PDC concepts as students progress through levels, as shown in Table 2.

Table 1: Game Elements Addressing PDC Challenges.

Challenge	How Addressed in Module
Difficult Material	1. Customizable technical levels 2. Presentation of concepts suitable for an introductory CS course
Instructor Backgrounds	1. Customizable technical levels 2. Inclusion of instructor materials
Specialized Hardware	1. Offered online
Student Lack of Interest	1.Based around a game 2.Includes interactive activities and assignments
Limited Instructional Time	1.Autonomous student use 2.Include pre- and post-game discussions 3.Incorporate all activities, programming assignments and assessments as appropriate

1.2 Gender Diversity

At the same time CS education is wrestling with the surge in the need for graduates with PDC skills, it is also attempting to overcome

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635591

a deficiency in diversity in CS including a striking gender disparity. Overall there is a general consensus that despite the efforts over the last few decades to lessen the gender imbalance in CS, the proportion of women is lower than it was in the 1980-1990s. The percentage of bachelor's degrees awarded to women overall in the US in 2021 [6] was only 19%, down from a high in the mid 1980-1990s of 37% [4]. There are many benefits to having a more diverse student population and a more balanced gender representation [7, 8, 13]. Therefore, it is necessary to find ways to encourage women to enter and stay in CS. Creating new educational materials provides just such an opportunity. To ensure the appeal of Parallel Islands to both women and men, a survey of the video game preferences of CS students was conducted. The results; such as a preference by all gender identities for strategy, cooperative, and goal- oriented games, aided the design of Parallel Islands.

1.3 Parallel Island

The gameplay portion of Parallel Islands is multi-player where the student works with other students (or BOTs) to complete tasks which introduce and demonstrate PDC concepts. The game further reinforces PDC concepts by showing a simplified, simulated version of what occurs behind the scenes of the multi-player game. This is accomplished by each player (or BOT) being represented by a thread and providing a visual representation of the status and activity of all the threads as shown in Figure 1. In addition to the game, each level of the module includes both a pre-level concept introduction and a post-level review with discussions, activities and programming assignments. The efficacy of Parallel Island was tested by administering basic PDC knowledge tests to student survey participants before and after gameplay. For most topics students scored 50% or less on the initial assessment and improved to 100% on the post-assessment. For the topics of critical sections and parallelism the post-assessment averages were slightly lower, at roughly 87% however the pre-test average was also lower (roughly 37%). The appeal of Parallel Island was determined by relevant post-gameplay survey questions. When asked to compare Parallel Islands to other educational games, 65% of students rated Parallel Islands higher.

Table 2: Parallel Concepts Covered per Gameplay Level.

	GamePlay	Concepts
Level 1	1. Solo Play 2. Speed-up: Part 1	1. Program 2. Process 3. Thread 4. Processors
Level 2	1. Multi-player (or BOTs) 2. Speed-up: Part 2 3. Correct Parallel execution	1. Speed-up 2. Concurrency 3. Parallelism 4. Time slicing
Level 3	1. Multi-player(or BOTs) 2. Parallel execution without mutual exclusion	1. Non-Determinism 2. Critical section 3. Mutual exclusion 4. Race Conditions 5. Shared Memory

Students also unanimously indicated they preferred learning PDC topics with Parallel Islands to textbook/lecture presentations of the material. However only half the students found the difficulty level appropriate which may be a reflection of the varied CS backgrounds of the participants.

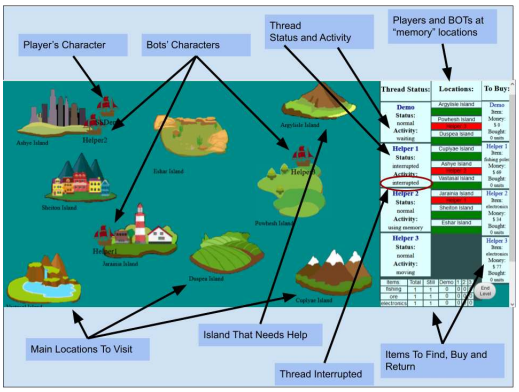


Figure 1: Game Board Showing Thread Status and Activity.

REFERENCES

[1] ABET. 2022. *ABET Criteria for Accrediting Engineering Programs, 2022 – 2023.* ABET. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2022-2023/.

[2] ACM/IEEE. [n. d.]. *Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* Association of Computing Machinery (ACM) and Institute of Electrical and Electronics Engineers (IEEE). https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf.

[3] Joel C. Adams. 2015. Patternlets: A Teaching Tool for Introducing Students to Parallel Design Patterns. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop.* 752–759.

[4] Carmen Botella, Silvia Rueda, Emilia López-Iñesta, and Paula Marzal. 2019. Gender diversity in STEM disciplines: A multiple factor problem. *Entropy* 21, 1 (2019), 30.

[5] L. Ivanov. 2012. The right balance: Restructuring the parallel and scientific computing course. *JCSC* 27, 3 (2012), 115—-121.

[6] Brian Kennedy, Richard Fry, and Cary Funk. 2021. *6 facts about America's STEM workforce and those training for it.* Report. PEW Research Center. https://www.pewresearch.org/short-reads/2021/04/14/6-facts-about-americas-stem-workforce-and-those-training-for-it/.

[7] Elizabeth Mannix and Margaret A Neale. 2005. What differences make a difference? The promise and reality of diverse teams in organizations. *Psychological science in the public interest* 6, 2 (2005), 31–55.

[8] Rose Niousha, Daisuke Saito, Hironori Washizaki, and Yoshiaki Fukazawa. 2023. Investigating the Effect of Binary Gender Preferences on Computational Thinking Skills. *Education Sciences* 13, 5 (2023), 433.

[9] Sushil K Prasad, Almadena Chtchelkanova, Sajal Das, Frank Dehne, Mohamed Gouda, Anshul Gupta, Joseph Jaja, Krishna Kant, Anita La Salle, Richard LeBlanc, et al. 2011. NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing: core topics for undergraduates. In *Proceedings of the 42nd ACM technical symposium on Computer science education.* 617–618.

[10] Erik Saule. 2018. Experiences on Teaching Parallel and Distributed Computing for Undergraduates. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW).* 361–368.

[11] Xiaoyuan Suo, Olga Glebova, David Liu, Alina Lazar, and Doina Bein. 2021. A Survey of Teaching PDC Content in Undergraduate Curriculum. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC).* 1306–1312.

[12] Neftali Watkinson, Aniket Shivam, Alexandru Nicolau, and Alexander Veidenbaum. 2019. Teaching Parallel Computing and Dependence Analysis with Python. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW).* 320–325.

[13] Chyanna Wee and Kian Meng Yap. 2021. Gender diversity in computing and immersive games for computer programming education: a review. *International Journal of Advanced Computer Science and Applications* 12, 5 (2021).

[14] Anthony Williams. 2012. *C++ concurrency in action.* London.

A Case Study of Taking AP Computer Science Principles: A Student's Perspective

Sarah Cameron
Department of Computer Science
University of West Florida
Pensacola, FL, USA
sec80@students.uwf.edu

Tony Pham
Department of Computer Science
University of West Florida
Pensacola, FL, USA
hdp12@students.uwf.edu

Dr. Sikha Bagui
Department of Computer Science
University of West Florida
Pensacola, FL, USA
bagui@uwf.edu

ABSTRACT

With the increased demand for Computer Science degrees in the work force, Computer Science is becoming more prominent in high schools. AP Computer Science Principles (AP CSP) is a course that serves as a bridge into Computer Science. Code.org provides a year-long curriculum for this AP course to be led by teachers in the classroom. Beyond an analysis of the pass rates of students, and with the recency of the AP CSP course, a reflection of the AP CSP curriculum from the student's perspective is in order. This study breaks down the strengths and weaknesses of AP CSP from a student's perspective. Results show there are many strengths compared to weaknesses in relation to the Code.org curriculum. However, the course can be a little challenging in motivating and engaging students if not executed properly by the teacher.

ACM Reference format:

Sarah Cameron, Tony Pham and Sikha Bagui. 2023. A Case Study of Taking AP Computer Science Principles: A Student's Perspective. In *Proceedings of 2024 Special Interest Group Computer Science Education (SIGCSE'24), March 20-23, Portland, OR.* ACM, Portland, OR, USA, 3 pages. https://doi.org/10.1145/3626253.3635490

1 INTRODUCTION

The field of computer science (CS) is a growing field, in need of graduating students with CS degrees to fill positions. In Florida alone, there were 28,088 open positions with only 3,808 graduates in 2022 [1]. To encourage students to pursue CS degrees at the college level, CS education is being introduced to students at the high school level. Many high schools are introducing CS courses, one version being the Advanced Placement (AP) Computer Science Principles (CSP) course. AP CSP is a course offered to high schoolers that allows them to earn college credit and learn about CS in a broader scope. Due to the recency of the course, research relating to the course is just beginning to be published. One area of lacking research is insight into the student's perspective on the course and its content. This work intends to present a reflection of the AP CSP curriculum from the student's perspective.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03. https://doi.org/10.1145/3626253.3635490

2 BACKGROUND INFORMATION

The AP CSP course was first launched in the 2016 exam season with the intent of bridging the gap for underrepresented groups in computer science including women, African Americans, and Hispanics. This intent has shown mixed results. AP's own report showed positive increases in underrepresented student enrollment in AP STEM-related courses [2], but this contrasts with data showing that the AP CSP course compared to the AP Computer Science A (CSA) course correlated with fewer declared CS majors (16.9% versus 28.3% respectively) [3]. A state-wide study on the AP CSP course, conducted in the state of Maryland, saw a decrease in the offerings of AP CSA once AP CSP was introduced in many districts [4]. Overall, research shows a mixed conversation about the AP CSP course, with little to no research specifically focusing on the student's perspective on the course.

3 UNDERSTANDING CODE.ORG

The students related to this research were taught using the AP CSP Code.org curriculum, a website-based curriculum recognized by AP [5]. The curriculum incorporates the 5 major topics of the exam into a 10-unit course with a combination of coding and conceptual-centric units.

The Code.org curriculum employs various resources to help students and teachers understand the course including worksheets, multi-media presentations, videos, projects, and assessments. The coding-centric units also employ a model by Code.org called EIPM (Explore, Investigate, Practice, Make). This EIPM model is intended for teachers to build a strong foundation of knowledge with their students before slowly allowing students to take charge of each topic [6].

4 ANALYSIS

To assess the Code.org curriculum for AP CSP from a student's perspective, both strengths and shortcomings of the course will be highlighted. Alongside these strengths and weaknesses, a section discussing the students' direct experiences is included.

4.1 Strengths

4.1.1 Usage of Multimedia Presentations. Code.org uses multimedia presentations to engage, motivate, and develop students' understanding of the material. This style of presentation is beneficial for students as it can "stimulate cognitive aspects of learning… [and] increase student motivation. [7]" Code.org directly integrates AP CSP vocabulary into multimedia presentations. Studies have shown that multimedia integration of vocabulary have helped in learning and retaining more vocabulary [8, 9].

4.1.2 Skill-Building: Collaboration to Debugging, Skills of Computer Science. Debugging, pair programming, and general collaboration are all skills emphasized in the course through projects and lessons. Debugging and collaborative skills are both vital in the field of computer science. A study has shown that learning through debugging allows students to have a more thorough understanding of coding principles [10]. The pair programming model has been shown to improve the confidence of students and the quality of projects when used in classes [11].

4.1.3 Projects, Assessments, and Handouts: Skill Assessment Towards Success. Code.org has a heavy emphasis on hands-on projects, AP-relevant assessments, and assistive handouts. A longitudinal study showed that students can develop a conceptual understanding rooted in creative and deeper thinking through projects [12].

4.2 Shortcomings

4.2.1 Conceptual and Coding Units: An Issue of Pacing. One major issue seen throughout the course is the issue of pacing. Units focusing on concepts relating to CS including cybersecurity, networks, and global connection are slow in nature compared to the extremely fast-paced coding units. Units relating to coding are often taught so quickly that newer students are unable to keep up.

4.2.2 Projects and Assessments: Disconnected from Curriculum. The projects and assessments for some units can feel disconnected from the material taught through the presentations. This disconnect doesn't allow students to benefit from the creative learning style brought about by project-centric learning.

4.3 Student's Perspective

4.3.1 Resources and Presentations: The Good and The Bad. The resources provided by Code.org for teachers to use are extensive and intended to support even the newest computer science educators in helping their students. However, these resources can fail to be effective for students if teachers do not provide access to these resources.

4.3.2 The EIPM Model: A System Failed in Application. The EIPM Model in design is meant to encourage students to slowly take initiative in their learning pertaining to each coding topic. However, if teachers fail to embrace the model's intended principles of weaning students off teacher support with each topic, the model falls flat. Failure to embrace the model by students and teachers can also harm the learning environment for coding-related concepts.

4.3.3 Teacher Involvement: A Curriculum Made for the Engaged. The curriculum is thorough in its explanations, providing extensive notes and a tight-knit community for teachers to connect to, but oftentimes, teachers don't embrace these available sources.

5 CONCLUSIONS

In general, beneficial approaches to teaching are found through the AP CSP Code.org curriculum including multimedia presentations with vocabulary integration; skill building including debugging, pair programming, and collaboration; and emphasis on teacher involvement including guidance and feedback. Despite the strong course, there are still some weak areas of pacing and disconnected projects and assessments. A big part of the student feedback emphasized the strength of the course material and setup but the failure of the teachers to embrace the material.

Overall, the students agreed that the course was quite effective at the material it taught but only failed in application due to teachers not being able to embrace all the material available.

ACKNOWLEDGMENTS

This work has been supported by NSF grant no. 2122393 and by The University of West Florida's Office of Undergraduate Research.

REFERENCES

[1] Code.org. 2022. Florida 2022 State of CS report | CS advocacy. Retrieved 2022 from https://advocacy.code.org/stateofcs.

[2] Jeff Wyatt, Jing Feng, and Maureen Ewing. 2020. AP computer science principles and the computer science pipelines. (December 2023). Retrieved 2022 from https://apcentral.collegeboard.org/media/pdf/ap-csp-and-stem-cs-pipelines.pdf.

[3] Linda J Sax, Kaitlin N. S Newhouse, Joanne Goode, Max Skorodinsky, Tomoko M Nakajima, and Michelle Sendowski. Does AP CS Principles Broaden Participation in Computing? February 2020. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education.* 542-548. https://doi.org/10.1145/3328778.3366826

[4] Heather Killen, David Weintrop, and Megean Garvin. AP Computer Science Principles' Impact on the Landscape of High School Computer Science using Maryland as a Model. February 2019. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education.* 1060-1066. https://doi.org/10.1145/3287324.3287356

[5] Code.org. 2023. CS Principles | Code.org. Retrieved 2022 from https://code.org/educate/csp.

[6] Code.org. 2022. EIPM: A Short Introduction - Google Docs. Retrieved 2022 from https://docs.google.com/document/d/1ncil5b0yWAN4LCyOeXwYuNrNKEHtN4nmAd2o-_K5Psw/preview.

[7] Iryna, Kotiash, Iryna Shevchuk, Maksym Porysonok, Iryna Matviienko, Mykyta Popov, Vitalii Terekhov, and Oleksandr Kuchai. Possibilities of using multimedia technologies in education. June 2022. *International Journal of Computer Science and Network Security* 22, 6 (June 2022), 727-732. https://doi.org/10.22937/IJCSNS.2022.22.6.91

[8] Hamidreza Khiyabani, Behzad Ghonsooly, and Zargham Ghabanchi. Using multimedia in teaching vocabulary in high school classes. January 2014. *Journal of Advances in English Language Teaching* 2, 1 (Jan. 2014), 1-13.

[9] Bagui, S. Reasons for Increase in Learning with Multimedia. The Journal of Multimedia and Hypermedia, 7(1), (1999), 3-18.

[10] J.M. Griffin. Learning by taking apart: deconstructing code by reading, tracing, and debugging. September 2016. In *Proceedings of the 17th Annual Conference on Information Technology Education.* 148-153. https://doi.org/10.1145/2978192.2978231

[11] C. McDowell, L Werner, H. E. Bullock, J. Fernald. The impact of pair programming on student performance, perception and persistence. May 2003. In *Proceedings of the 25th International Conference on Software Engineering.* 602-607. httsp://doi.org/10.1109/ICSE.2003.1201243

[12] Dimitra Kokotsaki, Victoria Menzies, and Andy Wiggins. Project-based learning: A review of the literature. July 2016. *Improving Schools* 19, 3 (July 2016), 267-277. httsp://doi.org/10.1177/1365480216659733

Exploring Collaboration as a Bridge to K-8 CS Integration

Justin Cannady
Northern Lights Collaborative for Computing Education
University of Minnesota
Minneapolis, MN, USA
canna053@umn.edu

Jennifer Rosato
Northern Lights Collaborative for Computing Education
University of Minnesota
Minneapolis, MN, USA
jrosato@umn.edu

Paul Schonfeld
Northern Lights Collaborative for Computing Education
University of Minnesota
Minneapolis, MN, USA
schon172@umn.edu

ABSTRACT

The landscape of computer science (CS) education in the United States has experienced a significant shift, with a notable emphasis on moving beyond introducing CS courses at the high school level to providing K-8 CS learning experiences. Early exposure to CS fosters interest and the foundational knowledge and skills needed for students to pursue future CS opportunities. However, the concept of equitable access to CS education at the elementary and middle school levels is still relatively nascent. To address the issue, we provide an in-depth analysis of our collaborative endeavor with a local public school district to develop and iteratively refine a professional development experience to enable teachers to implement a CS-integrated lesson in their classrooms. In this paper, we expand on the structure of the educator training experience and highlight strategies for curating and adapting CS lessons. In addition, we critically reflect on the strengths and challenges of this program based on teacher surveys and focus groups, as well as propose potential iterations for the program.

CCS CONCEPTS

• **Social and professional topics → K-12 education**.

KEYWORDS

K-8 computer science, integrated computer science, professional development, school district pathways

ACM Reference Format:
Justin Cannady, Jennifer Rosato, and Paul Schonfeld. 2024. Exploring Collaboration as a Bridge to K-8 CS Integration. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635565

1 INTRODUCTION

The rising demand for computational thinking and computer science (CT/CS) skills in the modern world necessitates their integration into K-12 education. This poster presentation summarizes our experiences in developing a professional development (PD) program for a local public school district. The program aimed at empowering teachers to incorporate CT/CS lessons into their curriculum. We discuss the structure of the educator training, and strategies for curating and adapting CT/CS lessons, and reflect on the strengths and challenges that emerged from this initiative.

2 BACKGROUND AND RELATED WORK

The shift in focus of K-12 CS education in the U.S. from high school to elementary and middle school has been supported by research suggesting the benefits of early exposure to CT/CS. However, challenges persist, including a lack of teacher training and resources, limited access to technology, and a scarcity of integrated CS curriculum [4]. Additionally, the concept of equitable access to CS education is still in its nascent stages, with certain populations being underserved due to a variety of factors [1]. While the integration of CT/CS into the curriculum offers a chance to foster computational thinking skills and promote digital literacy from an early age [5], it also requires carefully crafted lessons that are accessible to all students while providing sufficient coverage of both CT/CS and non-CT/CS academic learning objectives [3]. In addition, K-8 teachers have been reluctant to include computer science, viewing it as additional content, adding more to an already packed curriculum [6]. Along with low buy-in, educators report that they have a limited understanding of creating content that effectively integrates CT/CS into their classrooms [5].

To address these challenges, we partnered with a local public school district to develop a PD program for teachers. This program aimed to equip teachers with the knowledge and skills needed to integrate CS lessons into their existing curriculum, with the goal of increasing student access to inclusive CS education within the district. Prior work includes the CS4DLH project, initiated in 2021. CS4DLH aimed to increase access to computer science learning for students in Duluth, MN. The project was launched during the 2021 CS Ed Week with a PD event for elementary and middle school educators. In spring 2022, the project expanded to directly engage K-12 students and their families in City. The latest phase (school year 2022-23) of the project was designed to reach approximately 6,100 elementary, middle, and high school students and families.

3 DESIGN AND IMPLEMENTATION

Our approach to the program design and implementation was centered around co-designing CT/CS-integrated lessons with teachers. This collaborative process leveraged the expertise of educators in understanding the curriculum and the pedagogical approaches that work best for their students and community.

An initial meeting was held in the fall with lead teachers to identify potential topic areas and standards for integration in each subject. We then curated instructional materials that could serve as

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635565

a starting place for developing pedagogically sound lessons. The aim was to cultivate lessons that could be used in PD to provide a solid starting place for implementing a CT/CS-integrated lesson in classrooms. To ensure that the cultivated lessons aligned with state standards and were inclusive and engaging for all students, we developed a curriculum review rubric. The first draft of the curriculum review rubric was synthesized from the Teacher Accessibility, Equity, and Content (TEC) Rubric for Evaluating Computing Curricula [8] and the Evaluative Rubric for CT Integrated Curriculum in the Elementary Grades [7].

The initial set of lesson plans was reviewed by lead educators in a hands-on session several weeks before the PD. Through discussion between Organization team members and lead educators, a final lesson of approximately 2-4 days was selected for each subject area. Areas for revision and clarification were noted and then lessons were created following a template that aligned to the checklist.

The PD was co-designed with lead educators from the district, following guidelines for effective PD outlined by Darling-Hammond et al. [2]: 1) content-focused, 2) active learning, 3) models pedagogy, and 4) collaborative. This approach ensured teachers were engaged in learning, had opportunities to explore computing resources, and had a successful first programming experience that would give them the confidence to implement lessons in their classroom.

Two 3-hour PD sessions were offered after school in late fall. The first session focused on an introduction to computer science and computational thinking and the second session focused on inclusive pedagogy and teaching integrated CT/CS lessons. The second session of the PD introduced participants to the curated lesson plan in their subject areas and created time for lesson exploration, discussion, and adaptation. Participants gathered in subject-area groups and reviewed lessons for their grade level and subject area. After reviewing the lessons, teachers were guided through the lessons by the PD facilitators. The final activity allowed participants to discuss the implementation of the lessons and document changes and materials needed for implementation.

PD facilitators were available for teachers while implementing lessons, providing support in preparing for the lesson or being in the classroom during the lesson. Some teachers made additional adaptations to lessons during implementation. All teachers implemented lessons within five months of completing the PD. In late spring, after lesson implementation, teachers from each subject area gathered to reflect on what went well, what could be improved, and what else they would like to learn. Some participants also shared how they adapted lessons and provided examples of student work.

This approach of co-designing integrated lessons and providing comprehensive PD was instrumental in empowering teachers to incorporate CT/CS lessons into their curriculum. It not only provided them with the necessary knowledge and skills but also instilled a sense of confidence and enthusiasm in teaching CT/CS.

4 RESULTS AND DISCUSSION

Participants included 46 teachers across various disciplines and grade levels including 13 art specialists, 15 high school science teachers, 12 middle school math teachers, 5 elementary media specialists, and an 8th grade English language arts teacher. Survey data indicates positive impacts on participant perceptions, understanding, and readiness to integrate CT/CS into their classrooms. The program communicated the importance and relevance of CT/CS education, provided foundational CT/CS knowledge, and offered practical experience with CT/CS-integrated lessons. However, our data also highlight areas for improvement. For instance, while participants felt prepared to implement the lessons as they were initially presented, there is a clear need for more support and resources to help them tailor the lessons to their specific classrooms and academic disciplines.

In future iterations of the program, we propose greater emphasis on differentiated PD that caters to the unique needs and contexts of different subject areas. This could involve developing subject-specific resources and providing more targeted support, particularly for teachers who showed lower levels of confidence in their ability to teach CT/CS. Moreover, we propose to include additional scaffolded support for lesson adaptation, to empower teachers to tailor CT/CS-integrated lessons to their specific classrooms and academic disciplines. Also, by increasing the length and frequency of the PD with subsequent support opportunities, participants would continue to develop a higher level of competency and comfort in teaching computing concepts.

Based on participant feedback, we plan to revise the lessons to address identified challenges, including providing more scaffolded approaches for complex CS concepts, providing additional resources for extending the lessons, and offering more hands-on practice. We also plan to work with the school district to address the technical issues that were reported, such as web filtering and connectivity issues, to ensure a smoother implementation of lessons.

ACKNOWLEDGMENTS

This project was supported by the National Science Foundation (Award 1923369), CS4forLL, the Infosys Foundation USA, and ISD709 and would not have been possible without the engagement of district teachers.

REFERENCES

[1] Code.org, CSTA, and ECEP Alliance. 2022. 2022 State of computer science education: Understanding our national imperative.
[2] L. Darling-Hammond, M. Hyler, M. Gardner, and D. Espinoza. 2017. *Effective teacher professional development.* Learning Policy Institute, Palo Alto, CA.
[3] K. Fisler. 2021. Evolving a K-12 curriculum for integrating computer science into mathematics. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education.* ACM, 59–65. https://doi.org/10.1145/3408877.3432546.
[4] M. Israel. 2021. Understanding barriers to school-wide computational thinking integration at the elementary grades: Lessons from three schools. In *Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions*, A. Ottenbreit-Leftwich and A. Yadav (Eds.). ACM and the Robin Hood Learning + Technology Fund, 64–71.
[5] A. Ottenbreit-Leftwich and A. Yadav (Eds.). 2022. *Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions.* ACM and the Robin Hood Learning + Technology Fund, New York, NY, USA. https://doi.org/10.1145/3507951
[6] K.M. Rich, A. Yadav, and C.V. Schwarz. 2019. Computational thinking, mathematics, and science: Elementary teachers' perspectives on integration. *Jl Technol. Teach. Educ* 27, 2 (2019), 165–205.
[7] F.R. Sullivan, L. Duan, and E. Pektas. 2022. Design of an evaluative rubric for CT integrated curriculum in the elementary grades. In *6th International Conference on Computational Thinking Education and STEM Education (CTE-STEM.* https://doi.org/10.34641/ctestem.2022.475.
[8] D. Weintrop, M. Coenraad, J. Palmer, and D. Franklin. 2020. The teacher accessibility, equity, and content (TEC) rubric for evaluating computing curricula. *ACM Trans. Comput. Educ* 20, 1 (2020), 1–30. https://doi.org/10.1145/3371155.

On a Path to Indigenization and Decolonization of Computer Science Teaching

Sarah Carruthers
Vancouver Island University
Nanaimo, BC, Canada
carruthes2@viu.ca

ABSTRACT

In response to the Truth and Reconciliation Commission's Calls to Action, Canadian post-secondary institutions face the important, and daunting, task of decolonizing or Indigenizing curriculum across all disciplines. Computer science (CS) is not exempt from this responsibility and CS educators find themselves facing a challenging task. The field of computer science can seem very removed from Indigenous Ways of Knowing; training as an academic leaves many unprepared for the challenge of Indigenization or decolonization of curriculum or pedagogy; non-Indigenous educators are often deeply concerned with how to approach this task respectfully and authentically; and many computer science professors have had few opportunities to engage with Indigenous scholars and communities.

This poster explores four themes on a journey towards *preparing* for authentic Indigenization of CS curriculum. The journey begins with a reflection on the settler-colonial context in which many CS educators find themselves. Indigenization includes respectfully acknowledging the traditions, peoples, and cultures of the lands on which we live and work, which have largely been silenced by the colonization process. Authentic Indigenization should include building reciprocal relationships with Indigenous communities. Finally, CS educators will benefit from an interdisciplinary approach when exploring how to respectfully adopt and incorporate different worldviews into their teaching practice. This poster provides an opportunity to reflect upon and discuss these important facets of Indigenizing and decolonizing our curriculum.

ACM Reference Format:
Sarah Carruthers. 2024. On a Path to Indigenization and Decolonization of Computer Science Teaching. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635504

1 FURTHER INFORMATION

The purpose of this poster is not to describe *how* Computer Science (CS) curriculum can be Indigenized or decolonized, but instead to provide a narrative of how one CS educator is approaching preparing for Indigenization of their teaching. The poster reflects the personal nature of the journey on the path towards Indigenization, with the goal of providing a starting point for conversations about the challenges and rewards encountered so far on this path. It also aims to provide some helpful resources[1] for getting started with Indigenization work as a stepping stone for other educators to take first steps of their own.

In recent years, great efforts have gone into recognizing and respecting the rights of Indigenous peoples, as described in the United Nations Declaration of the Rights of Indigenous Peoples (UNDRIP) [15], and, in Canada, the Truth and Reconciliation Commission's Calls to Action [4]. These documents aim to acknowledge the rights of Indigenous Peoples, and guide steps that can be taken to uphold these rights and achieve reconciliation. Educators and educational institutions play a central role in reconciliation, by ensuring that Indigenous rights are recognized and respected by students in their programs. As a result, there is a growing movement to Indigenize and decolonize education across Canada, to facilitate the understanding and respect of Indigenous rights and culture. How, and to what extent, educational institutions are choosing to Indigenize and decolonize their courses, programs, and institutional practices varies, and so as educators we find ourselves in a very diverse landscape. As examples in Western Canada, Vancouver Island University, like many institutions, has updated their strategic plan [20] to address the importance of Indigenous knowledge, rights, and culture; the University of British Columbia has developed an Indigenous Strategic Plan [16] that outlines more specifically how they plan to work towards reconciliation; and the University of Fraser Valley requires all curriculum, institution-wide, to be Indigenized [17].

Instructors and curriculum designers are faced with determining how to Indigenize their curriculum and practices. These efforts are supported by institutional Teaching and Learning Specialists, who offer workshops and support for the process of Indigenization. Respecting the integrity of Indigenous knowledge, many institutions include Indigenous Elders and Knowledge Keepers who can support and guide instructional faculty [19, 21] on their journey towards Indigenization. Some institutions also provide curated lists of resources to help instructors approach Indigenization [13, 22].

Despite these supports, some fields of study, like computer science, still seem to be particularly challenging to Indigenize. It may not be immediately evident how Indigenous Ways of Knowing, or the history of Indigenous peoples in Canada, might be incorporated into undergraduate CS curriculum. This is reflected in the lack of publications on Indigenization or decolonization of CS curriculum, worldwide. While a preliminary search for conference and journal publications yields only a few results directly related to this work, the good news is that there are early works in decolonization of CS curriculum that provide starting points for those wishing to

[1] Available online here: http://csci.viu.ca/~carruths/SIGCSE2024Sources.html

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635504

begin [2, 6, 7]. This limited prior work on Indigenization and decolonization of CS means that curriculum designers and instructors will benefit from broadening their search to related disciplines with more experience in this area, such as Mathematics which is ahead of our discipline in this regard [1, 5, 10, 14, 18]. Core to Indigenization and decolonization is being open to worldviews other than traditional western scientific worldview. CS educators will need to be open to new and varied perspectives, to get any traction in this process.

My path to Indigenization began at a workshop introducing the 4Rs (Respect, Reciprocity, Responsibility, and Relevance) as a framework for Indigenization of curriculum. The 4Rs were first coined by Cree scholar Verna Kirkness and Ray Barnhardt in the early 1990s as a way for educational institutions to assess the extent to which they were meeting the needs of Indigenous students [9]. The 4Rs, and variants, are now commonly referenced by scholars working to make their teaching more accessible to Indigenous students [3, 11, 12], and seemed to be an appropriate example of Indigenous Knowledge to start with on this journey. In September 2022, I had the opportunity to share preliminary thoughts of how to include the 4Rs in my CS teaching with Verna Kirkness over Zoom, where I was met with keen questions and sincere enthusiasm for the pursuit of Indigenizing CS curriculum in this manner [8]. While this journey continues to be fraught with self-doubt and concerns I will make missteps, when in doubt, I think back to Verna's unbounded enthusiasm for making STEM, and CS in particular, more accessible to Indigenous students. I look forward to working to make my classroom more accessible to Indigenous students through the use of the 4Rs. I reflect on the value future CS students will gain from being introduced to new worldviews and ways of knowing in my classes. I look forward to conversations, both enthusiastic and challenging, about how we as CS educators can adapt our discipline to be more accessible and open to worldviews and knowledge beyond what we already are comfortable with.

Indigenization is in part a personal process, which is why I chose to end with this personal reflection, one that aligns with the narrative format of the poster. The authentic practice of Indigenization requires the recognition and adoption of new worldviews, and becoming comfortable with new ways of knowing. As such, CS instructors and curriculum designers working towards Indigenization and decolonization will find ourselves on slightly different (and often personal) paths, depending on who we are, what knowledge we carry, where we live, and who we build relationships with. With this poster I share my journey thus far, recognizing that I (we) have only just begun.

ACKNOWLEDGMENTS

To Cree scholar Verna Kirkness, for her ongoing and inspiring dedication to Indigenous STEM engagement and education, whose work inspired this journey.

This work is funded by VIUFA Assisted Leave Award.

REFERENCES

[1] Piata Allen and Tony Trinick. 2022. Creating space for indigenous knowledge in Māori-medium mathematics classrooms. *International Journal of Qualitative Studies in Education* (2022), 1–14.

[2] Dianne Biin and Marla L Weston. 2015. An Indigenous Learning Approach to Computer Science Education: 21st Century Skills for Middle and High School Aboriginal Children on British Columbia's West Coast. In *Emerging Technologies for STEAM Education*, Xun Ge, Dirk Ifenthaler, and J. Michael Spector (Eds.). Springer International Publishing, Cham, 95–112. https://doi.org/10.1007/978-3-319-02573-5_6

[3] Candace Brunette-Debassige and Pauline Wakeham. 2020. Reimagining the Four Rs of Indigenous Education for Literary Studies: Learning From and With Indigenous Stories in the Classroom. *Studies in American Indian Literatures* 32, 3-4 (2020), 13–40. https://doi.org/10.1353/ail.2020.0016

[4] Government of Canada; Crown-Indigenous Relations and Northern Affairs Canada. 2015. Truth and Reconciliation Commission of Canada. https://rcaanc-cirnac.gc.ca/eng/1450124405592/1529106060525 Last Modified: 2022-09-29.

[5] Shana RW Graham. 2015. Indigenization of mathematics curriculum: An evolving experience. In *Mathematics Education and Society Conference*. 170.

[6] Mawera Karetai, Samuel Mann, Dhammika Dave Guruge, Sherlock Licorish, and Alison Clear. 2023. Decolonising Computer Science Education - A Global Perspective. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1097–1102. https://doi.org/10.1145/3545945.3569870

[7] Mawera Karetai, Samuel Mann, Dhammika Dave Guruge, Sherlock Licorish, and Alison Clear. 2023. Social Dreaming Together - Envisioning Decolonised Computer Science Education. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1224–1225. https://doi.org/10.1145/3545947.3569597

[8] Verna J Kirkness. 2022. Personal Conversation.

[9] Verna J Kirkness and Ray Barnhardt. 1991. First Nations and higher education-The Four R's Respect, Relevance, Reciprocity, Responsibility. *Journal of American Indian Education* (1991), 1–15.

[10] Vimolan Mudaly. 2018. Decolonising the mind: Mathematics teachers explore possibilities for indigenising the school curriculum. *Journal of Education (University of KwaZulu-Natal)* 74 (2018), 67–84. https://doi.org/10.17159/2520-9868/i74a05

[11] Karen Ragoonaden and Lyle Mueller. 2017. Culturally Responsive Pedagogy: Indigenizing Curriculum. *Canadian Journal of Higher Education* 47, 2 (2017), 22–46.

[12] Jillian Seniuk Cicek, Alan Steele, Sarah Gauthier, Afua Adobea Mante, Pamela Wolf, Mary Robinson, and Stephen Mattucci. 2021. Indigenizing Engineering education in Canada: critically considered. *Teaching in Higher Education* 26, 7-8 (Nov. 2021), 1038–1059. https://doi.org/10.1080/13562517.2021.1935847

[13] Simon Fraser University. 2023. Indigenizing curriculum | SFU Library. https://www.lib.sfu.ca/help/academic-integrity/indigenous-initiatives/icrc/indigenizing-curriculum

[14] Brian Tweed. 2021. The doing of curriculum mathematics: the case of an Indigenous Maori school in Aotearoa/New Zealand. *British Journal of Sociology of Education* 42, 5-6 (Aug. 2021), 914–931. https://doi.org/10.1080/01425692.2021.1941764

[15] United Nations. 2007. United Nations Declaration on the Rights of Indigenous Peoples. https://social.desa.un.org/issues/indigenous-peoples/united-nations-declaration-on-the-rights-of-indigenous-peoples

[16] University of British Columbia. 2021. Indigenous Strategic Plan. https://indigenous.ubc.ca/indigenous-engagement/indigenous-strategic-plan/

[17] University of the Fraser Valley. 2023. Xwexwílmexwawt > Indigenization at UFV. https://www.ufv.ca/indigenous/indigenization/

[18] Niken Wahyu Utami, Suminto A. Sayuti, and Jailani Jailani. 2021. Indigenous artifacts from remote areas, used to design a lesson plan for preservice math teachers regarding sustainable education. *Heliyon* 7, 3 (March 2021). https://doi.org/10.1016/j.heliyon.2021.e06417

[19] University of the Fraser Valley. 2023. Teaching and Learning Centre > Inclusive teaching > Indigenization. https://ufv.ca/teaching-and-learning/inclusive-teaching/indigenization/

[20] Vancouver Island University. 2021. People Place Potential Mukw'lhwet q'ushin'tul tu shniis ttha tat-ulut mustimuxw: Strategic Plan 2021-2026. https://gov.viu.ca/sites/default/files/viu-strategic-plan.pdf

[21] Vancouver Island University. 2023. Elders at VIU. https://indigenous.viu.ca/office-indigenous-education-and-engagement/elders-viu

[22] Martin Warkentin. 2023. LibGuides UFV: Indigenizing the Academy: Weaving Knowledge Systems Resource Materials. https://libguides.ufv.ca/IndigenizingtheAcademy/weavingknowledgesystemsresourcematerials

Embracing Mathematical Conjecture Through Coding and Computational Thinking

Sarah D. Castle
scastle@uidaho.edu
University of Idaho
Moscow, Idaho, USA

ABSTRACT

Mathematics and computer science education have an intertwined history as proof undergirds both disciplines. This initial instrumental case study follows 16 different computational science students through a series of 8 linear algebra modules. Thematic analysis was used to highlight that when students learn mathematics through computation, opportunities for conjecture can be fostered alongside computational thinking.

ACM Reference Format:
Sarah D. Castle. 2024. Embracing Mathematical Conjecture Through Coding and Computational Thinking. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635561

1 INTRODUCTION

Mathematics courses are an integral part of many computer science programs; however, some question mathematics courses' effectiveness [1, 2, 8]. In response, researchers have examined the intersection of computing and mathematics [7, 9], and leveraging CS students' strengths, namely computing, as a pedagogical approach to mathematics [3]. An underlying foundation of mathematics is proof and conjecture, where a conjecture is a mathematical statement or pattern that is believed to be true but has not been proven yet. However, proof is also critical in computer science, particularly when relating to programs and algorithms; yet the research and education community lacks how computing can support proof and vice-versa [5]. The presented study investigated this relationship and asked the question: *How can coding and computation support students in mathematical conjecture?* These preliminary results will hopefully spark discussion about how to support CS students in their mathematics courses by leveraging a familiar environment.

2 THEORETICAL PERSPECTIVES

Within literature, technology has promoted cycles of proof where students use technology to explore a mathematical topic, generate their conjectures, test/revise the conjectures, and finally prove them [4]. Mathematics itself is not a static process but rather a dynamic process of conjecturing, discovery, proof and refutation, something often missing from mathematics when presented as definition-theorem-proof [6]. Therefore, technology has the potential to shift student action to allow for exploration and conjecturing. However, much of the research centers on providing students with a tool and then having them explore. Students are not in the creation of the tool, nor are they able to modify it. Coding on the other hand allows for students to explore a tool, modify it, or even create their own. Therefore, there is potential for students to engage in authentic mathematical conjecturing that places them as the source of mathematical authority.

3 METHODS

This study uses an instrumental case study approach where the issue was computation fostering opportunities for mathematical proof, specifically conjecturing. This was a case of a set of enacted materials where students engaged in conjecture aided by computing. This study followed 16 participants recruited from an introduction to computational modeling course as they engaged in 8 different linear algebra modules developed in the Jupyter notebook environment using Python [3]. Students were put into small groups of 3-4 students and met for 2 hours every week for a total of 8 weeks with accompanying audio, video, and individual screen recording data. Each student completed an initial interview and pre-study survey, weekly reflections after each linear algebra module, and a final interview and post-study survey. All data sources were coordinated, transcribed, and coded along the dimensions of the proof cycle previously noted. The conjecturing codes were grouped and coded for mediation by the coding, and a thematic analysis identified ways in which computing and coding supported conjecture.

4 PRELIMINARY RESULTS

An initial claim from this analysis was students used coding to engage in the practice of conjecturing through experimentation. Further, through the coding of mathematics - opportunities for computational thinking arose, specifically through decomposition, abstraction, generalization, and problem solving. One of the key affordances of coding itself was that students could use the code to test multiple cases of their conceptualization. They simply changed a parameter without redoing all of their work. When discussing his experience in this study, Jack reflected on coding, "I think it helped with it being iterable where we could just try this idea and this idea and this idea instead of having to take the five minutes in between to erase everything, write it out again, do all the steps". Other students expressed similar views stating that the ability to run code with new parameters without having to redo everything allowed them to "play around and try new ideas and see how things – if they work the way you thought they would". This was crucial as it allowed for the generation of different data where students

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635561

could look for patterns and abstract a mathematical idea or pattern. Students started with specific tests and cases to then create a more generalized conjecture. For example, consider Micah's refection when asked what helped push his mathematical thinking:

> The examples helped push it a lot or, when it came to like the cat images or transforming things or changing those matrices. It was always looking at what happened whenever I changed the matrix and then always thinking back on why it happened almost - so I'd always try and just play around with matrices until finding something that worked... If you change the value from zero to one and then one to negative one or something, whatever that did to the image. I just broke it down to the steps about, well, if I change the number here, what does it do? Like the overall equation. I had to do several trials to find out what value in the matrix changed and the graph or something... I first try to figure out matrices and solve them because I just have to guess a little bit about what's going on and then eventually, I'll put it together. So usually, I could piece it together towards the end.

In this excerpt, Micah highlighted how his process involved making modifications and then looking at the effect on the visual representation that he created within Python. Specifically, he investigated how a transformation matrix impacts a points matrix. He noted the ability to run several trials was key to him being able to piece together his conception of matrices. Further, during this process he reflected on why something happened until he was able to come up with a general conjecture that seemed to fit his computational trials. In doing this process, students engaged in the aforementioned proof cycle through exploring mathematical ideas, conjecturing, and testing and refining their conjectures. Further, during this process one of the affordances was that students engaged in generalization and abstraction. Micah abstracted general types of transformation matrices and their effects on images by breaking down the process and identify the key components that altered his images and deduce a pattern. He used his code to test more examples and built his conjecture. From here, he could modify his approach to investigate affine transformations and the order of linear transformations and conjecture about their properties. Because he had built the code for matrix transformations, he simply modified his code to test his initial conjecture. For example, with the order of linear transformations, he conjectured in his notebook:

> Linear transformations cannot be applied in any order, this is because the first matrix created already has all the values changed from the first transformation, whereas the second transformation has to use those values. So, if they were vise-versa it would augment the matrix differently.

He tested the order of different transformation matrices using code and refined his reasoning behind his conjecture until he was satisfied. The openness and adaptability of the coding environment allowed for mathematical experimentation and conjecture testing.

The act of decomposition was not isolated to Micah, as Alex noted that when working on the mathematics problems through coding he tried to "break it down a bit more into smaller parts". With regards to computational mathematics, Jack stated "It's easier to just work on one part at a time, individually, get those parts kind of working, and then put them all together and see if anything falls apart then or if you get a final product that you're looking for". This was seen across observations, as one of students' first steps when working on a complicated problem was to start at the board and write the pseudo code or a general outline to a problem. Harper and Theo highlighted the usefulness of breaking down a complicated problem on the board, parsing it into different steps, and being able to work on the different pieces. In observations students used this ability to start with a small trial and then expanded it while developing their conjecture. For example, when investigating the relationship between matrices and the corresponding determinant, students first wrote code to find the numerical determinant and then expanded to the visualization to relate linear independence and the value of the determinant. Additionally, they systematically investigated matrices and the determinant by starting with the identity matrix, shifting to other matrices, and then trying known matrices constructed with linearly dependent and independent vectors. The code and engaging in computing allowed them to experiment and explore to build up their conjecture.

5 DISCUSSION AND CONCLUSION

These findings provide an initial basis to consider how computation and coding could potentially support proof through engaging students in conjecturing through experimentation and exploration. Further, these opportunities fostered abstraction, decomposition, and problem solving thereby highlighting the potential pedagogical symbiotic relationship between computation and mathematics. Although technology has been seen as a way to prompt proof cycles [4], this work broadens the potential by fostering conjecture and computational thinking simultaneously. As technology continues to adapt and change, it is important that educators and researchers respond and continually strive to develop environments that support CS student learning through leveraging novel tools.

REFERENCES

[1] Douglas Baldwin, Henry M Walker, and Peter B Henderson. 2013. The roles of mathematics in computer science. *Acm Inroads* 4, 4 (2013), 74–80.

[2] Trudie Benadéa and Janet Liebenberga. 2019. The relevance of a Mathematics course for computer science students. In *Swan Delta Proceedings: The 12th Delta Conference on the Teaching and Learning of Undergraduate Mathematics and Statistics*. 2–10.

[3] Sarah D Castle. 2023. Leveraging Computational Science Students' Coding Strengths for Mathematics Learning. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 263–269.

[4] Craig J Cullen, Joshua T Hertel, and Megan Nickels. 2020. The roles of technology in mathematics education. In *The Educational Forum*, Vol. 84. Taylor & Francis, 166–178.

[5] Viviane Durand-Guerrier, Antoine Meyer, and Simon Modeste. 2019. Didactical issues at the interface of mathematics and computer science. *Proof technology in mathematics research and teaching* (2019), 115–138.

[6] Imre Lakatos. 1976. *Proofs and Refutations: The Logic of Mathematical Discovery*. Cambridge University Press. https://doi.org/10.1017/CBO9781139171472

[7] Elise Lockwood and Knut Mørken. 2021. A call for research that explores relationships between computing and mathematical thinking and activity in RUME. *International Journal of Research in Undergraduate Mathematics Education* (2021), 1–13.

[8] Anthony Ralston and Mary Shaw. 1980. Curriculum'78—is computer science really that unmathematical? *Commun. ACM* 23, 2 (1980), 67–70.

[9] Odd Petter Sand, Elise Lockwood, Marcos D Caballero, and Knut Mørken. 2022. Three cases that demonstrate how students connect the domains of mathematics and computing. *The Journal of Mathematical Behavior* 67 (2022), 100955.

Enhancing Automated Feedback in On-Going Assignments

Huanyi Chen
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada
huanyi.chen@uwaterloo.ca

Paul A.S. Ward
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada
pasward@uwaterloo.ca

ABSTRACT

In programming courses, test-based automated feedback systems often face a limitation: instructors cannot effectively enhance feedback during ongoing assignments. While a passing test case may adequately indicate that certain rubric criteria have been met by a student's program, failure can arise from many reasons. When a test case fails due to reasons not previously coded for, it can create confusion. This tends to divert a student's focus from genuine learning towards guessing the test. A more effective approach would be for the system to notify instructors when a test fails, rather than automatically returning a "test case failed" message. Instructors could then diagnose the cause to provide an initial "human-in-the-loop" feedback. Subsequently, this feedback can be coded into the test suite. This method enables the improvement of feedback during an on-going assignment. In this poster, we propose an approach to achieve this objective and introduce a supporting tool.

ACM Reference Format:
Huanyi Chen and Paul A.S. Ward. 2024. Enhancing Automated Feedback in On-Going Assignments. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635571

1 PROBLEM STATEMENTS

The automated feedback system is a vital component in programming courses. Instructors rely on it to deliver timely feedback to students on a large scale. While a passed test case often reasonably indicates that a student's program satisfies certain required rubric items, the converse is not typically true. When a test case fails, the cause can be unforeseen. For example, if a test writer assumes the execution result of the student's program is always numeric, a line to cast the result into integers might be used. However, if a student forgets this rule, the test case will fail due to invalid casting. Given that this is unexpected, it is unlikely that this test can provide correct feedback. This can be particularly challenging for students who are already unsure about their solutions.

Adding to the complexity, instructors usually find it difficult to modify or correct the assessment code during an ongoing assignment. This is because changes to the assessment code could lead to students receiving inconsistent feedback for the same solution submitted at different times. Consequently, instructors often postpone addressing these issues until the assignment concludes, implementing any improvements in subsequent offerings of the same assignment [1].

Just as assessments often require modifications, assignments too frequently need revisions, especially newly designed ones. As a result, new assessment code is introduced, and the aforementioned issues re-emerge.

The problems can be roughly summarized as:

(1) **Unclear failure cause**: The assessment code offers limited insight into the actual reason behind a failed test case.
(2) **Inconsistencies**: Modifying the assessment code can lead to inconsistent feedback messages.

In this poster, we propose an approach that allows instructors to intervene when unexpected situations arise during assessment. This approach also empowers instructors to enhance the assessment code without waiting for assignments to conclude. We open-source a supporting tool to fulfill the idea. We hope for it to find broader adoption.

2 PROPOSED SOLUTIONS

Figure 1 shows the overview of our solutions.

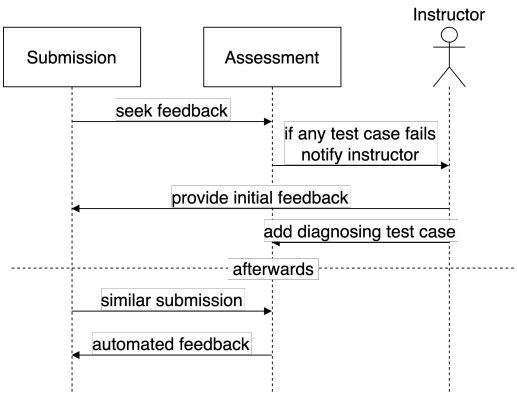


Figure 1: Enhancing automated feedback procedure

2.1 Unclear failure cause

The rationale behind our solution for this problem is that, a passed test case indicates all code of the test case has executed as expected, while a failed test case can fail due to unforeseen causes, which may or may not be the location anticipated by the test case writer. Thus, the specific reason for such failures might not have been precoded.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635571

To address this, we consider the assessment code should only report if a program is correct, *i.e.*, all test cases passed, but it should not report if a program is incorrect, *i.e.*, some test cases failed. For the latter, it notifies instructors and tell students to wait for human-in-the-loop feedback.

Students anticipate feedback to correct misconceptions. Therefore, the assessment code should help diagnosing the root cause of test case failures. Instructors, upon inspecting a failed submission, can introduce a "diagnosing test case" to pinpoint the failure cause when it passes.

This iterative process continues until all root causes can be identified by diagnosing test cases.

2.2 Inconsistency due to assessment code changes

We propose that instructors should always translate test case results into actionable feedback to reduce the cognitive burden on students when interpreting test case outcomes. If a primary test case fails, instructors should reference the corresponding passed diagnosing test case to provide feedback.

Although instructors will not automatically be notified of positive test outcomes, they can detect false positive results through manual inspection. Since we do not disclose the outcomes of individual test cases to students, we can offer feedback such as, "*Your solution seems sound! However, we encourage you to think about potential edge cases that our current evaluation might miss.*" This approach fosters critical thinking among students, and helps students to separate feedback and grades. Receiving perfect feedback may not ensure a solution is perfect, however, it may indicate that the solution is on the right track and likely to gain a reasonable grade.

Nonetheless, we believe it is preferable not to introduce code changes that invalidate previously correct solutions in primary test cases, as this can cause insecurity among students, leading to a surge in resubmissions.

Modifications for above cases should be implemented after the assignment is concluded. In making these modifications, the approach used in Haldeman et al. [1] could be one of the options.

For false positives within diagnosing test cases, instead of immediately modifying the test case, we might consider adjusting the feedback to include, "*...or another possible issue could be ...*".

As for addressing false negatives, since we prevent students from seeing any feedback when any test case fails, therefore, we can make corrections as soon as we identify an issue.

3 THE SUPPORTING TOOL

We have developed a tool[1] designed to help instructors incorporate our ideas into their courses. The current version requires instructors to create test cases using the popular "pytest" testing framework. Although primarily built for Python, it can be configured to assess assignments of other programming languages as well. For instance, we are currently using it for database assignments.

The tool needs instructors to provide feedback messages with diagnosing test cases. If an automated feedback is available, the feedback is used; if not, an email is sent to the instructor. What are included in the email are configurable. It also allows the instructor to generate a draft AI feedback and include it in the email.

Currently, we consider a feedback message might need to be associated with one or more diagnosing test cases. Therefore, our tool supports matching a set of diagnosing test case outcomes to a feedback message. However, we do not have a strong rationale behind the assumption.

The tool is a command-line application that can be distributed, installed, and executed in various environments. Although the tool does not offer much assistance in writing assessment code, it provides a common structure for individuals to develop additional features for their assessment code and potentially share them, beyond what we have implemented, such as sending emails and generating AI feedback.

4 DISCUSSION

The quality of assessments should be the most essential component in automated feedback systems; however, this has been rarely discussed. Our approach outlines a procedure to improve the quality of assessments deterministically.

We are currently using the tool in a database course. In particular, we will primarily use it to assess students' Structured Query Language (SQL) queries. Beyond this, we have also developed encoding techniques to encode Relational Algebra (RA) and Entity Relational (ER) modelling solutions. We will explore how the tool helps for those questions as well.

We integrate our tool into an automated system where students submit their work. We found the tool is very helpful. For example, many students encountered a similar runtime error. We quickly added a diagnosing test case to detect it and provide an interpretation.

We had to spend time on monitoring the emails when it was close to the assignment deadlines. However, we believe instructors would inevitably have to invest the time if the automated feedback were not helpful anyways (which was also our experience).

Although we have found the AI can generate fairly accurate feedback on some types of mistakes, whether we can directly send back the AI feedback is still unclear (even though we have done so in cases such as syntax errors).

Given the AI generated feedback, it might seem to shift the focus of instructors from adding new tests to reviewing the AI feedback before sending it to students. However, our experiences show that, it is unclear what prompts should be used to generate high-quality AI feedback. We observed that sometimes the AI feedback leaks the canonical solution. However, we are unsure how to disallow it even though we have tried adjusting the prompts.

We will evaluate the efficacy of the tool by comparing students' performance with data from previous terms, wherein the tool was not deployed. We anticipate observing a marked improvement in students' proficiency in the relevant learning objectives.

REFERENCES

[1] Georgiana Haldeman, Monica Babeş-Vroman, Andrew Tjang, and Thu D. Nguyen. 2021. CSF: Formative Feedback in Autograding. *ACM Transactions on Computing Education* 21, 3, Article 21 (May 2021). https://doi.org/10.1145/3445983

[1] https://github.com/h365chen/socassess

Assessment-via-Teaching: Exploring an Alternative Assessment Strategy in Undergraduate Introductory Data Science Course

Lujie Karen Chen
lujiec@umbc.edu
University of Maryland, Baltimore County
Baltimore, MD, USA

Justin Thai
jthai2@umbc.edu
University of Maryland, Baltimore County
Baltimore, MD, USA

ABSTRACT

Learning-by-teaching is an active learning method that has the promise of engaging students and enhancing their learning. These benefits include the improvement of the metacognitive process, increased motivation and self-efficacy, and the opportunity to refine communication skills. These benefits are particularly valuable in data science education. Though not as widely studied, teaching assignments, where students demonstrate competency through teaching others can be used as a formative assessment tool. This paper describes an "alternative midterm" experiment conducted in an undergraduate introductory data science class. In this pilot, students were asked to demonstrate their newly learned data science skills by teaching them to individuals without data science backgrounds. The evidence of learning will be illustrated through an in-depth analysis of the students' teaching products. These products consist of two parts: (1) "Teaching Preparation" materials, such as Python Notebooks containing problems and solutions crafted by the students, and (2) "Teaching Implementation", video recordings of real-time interactions between the students and their "tutees." The paper will also present qualitative feedback from the students, highlighting promising signs of engagement and acceptance. We will conclude by discussing the challenges and opportunities of implementing this assessment strategy on a larger scale.

ACM Reference Format:
Lujie Karen Chen and Justin Thai. 2024. Assessment-via-Teaching: Exploring an Alternative Assessment Strategy in Undergraduate Introductory Data Science Course . In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635516

1 INTRODUCTION

Real-world data science requires proficiency in technical skills, such as computation and quantitative reasoning, and the ability to effectively communicate with domain experts and audiences unfamiliar with data science. The question arises: how can we accurately assess these essential skills? The learning-by-teaching approach is an active learning technique known for its multi-fold benefits. It enhances the metacognitive process, motivates learners, improves learning strategies, and provides opportunities to sharpen communication skills [1]. Assessment-via-Teaching, where students are asked to demonstrate their competency through teaching those without data science backgrounds, can be used as a formative assessment strategy, offering insights into students' multi-facet mastery, including content knowledge and communication skills. The computing education community has explored non-traditional assessment methods, such as analyzing recorded code-tracing videos [2], to gain a deep understanding of students' learning. This research contributes to the emerging literature on alternative assessment strategies for complex and rich assessment objectives involving technical skills such as programming and quantitative reasoning and soft skills to communicate to general non-technical audiences.

2 COURSE CONTEXT

This pilot took place in an introductory data science course at the undergraduate level in Fall 2022 at a 4-year Minority Serving Institute in the US. The curriculum was adapted from the Data 8 Introduction to Data Science from the University of California, University of California, Berkeley. This curriculum focuses on developing basic data science concepts in statistical inferences and computation at the same time. This course serves as the first introduction to data science, with almost no prerequisite. Students only need to have a math level of high school Algebra I, and no prior experience in programming is required. The course is open to students from all majors from across the campus. The adapted version is a three-credit course with two and a half hours of in-person instruction over 16 weeks. The first half of the course focuses on basic table functions and plotting functions, and the second half focuses on statistical inference using computational simulation. The teaching assignment described in this paper serves as the mid-term formative assessment of the material covered during the first half of the course. Enrollment is typically capped at 25 students and with the support of undergraduate teaching fellows who took the class before. Fall 2022 is the fifth offering of this course at this institute, with 25 students from a diverse range of majors across the campus and over 80% students are from minority backgrounds.

3 TEACHING ASSIGNMENT

Note: We use the following naming convention to avoid confusion. In the context of discussion of the course, the teacher of the course is called the "instructor", and the learner in the course is called the "student,"; while in the context of teaching assignment, the data science student who plays the teaching role is called "tutor" and his/her "student" is called "tutee".

Figure 1 gives an overview of the timeline of the teaching assignment involving four major steps.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635516

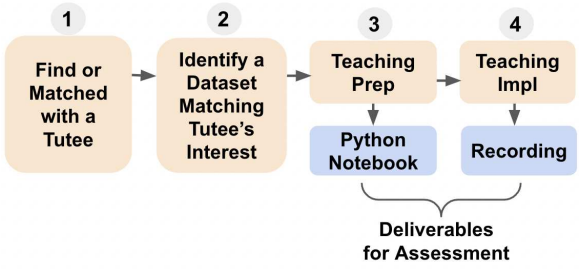


Figure 1: Overview of the Timeline of the Teaching Assignment

(1) **Find or Matched with a Tutee**. Tutees can be anyone from the broader campus community, neighborhood, or circle of friends, regardless of profession and educational background. The only criterion is that he/she does not have prior knowledge or experience with data science or programming. Students may find a tutee on their own or be matched to one by the instructor who maintains a list of undergraduate students outside data science disciplines who express interest in this activity. The matching is mostly random, though it might consider the common interests between the tutor and the tutee if information is available.
(2) **Identify a Dataset.**Students work with his/her tutee to identify a dataset that matches the tutee's interest. The instructor provides a list of publicly available dataset that could be used, but students are free to find dataset on their own. The instructor reviewed the selected datasets to ensure they were with appropriate content and level of complexity.
(3) **Teaching Preparation** Students develop teaching preparation materials in Python Notebook, which is the primary programming environment for the course. The regular assignments are hosted on a private JupyterHub at the institute. We use Google Colab Python Notebook to accommodate tutees from outside the class. A tutorial was provided to students to familiarize them with the Google Colab. To scaffold the teaching preparation, students were asked to propose five "curious questions" about the dataset (encouraged to work with a tutee to get those questions) and then write code to answer those questions. A class period was devoted to demonstrating the development of a teaching notebook in Google Colab and teaching using the notebook.
(4) **Teaching Implementation**. Students implement teaching with their tutees. The teaching modalities are flexible: students may teach online or in person. The lesson typically lasted between 30 to 45 minutes. Students are asked to record the lessons to be used for assessment.

4 ASSESSMENT STRATEGY AND FORMATIVE FEEDBACK

The assessment rubric comprises two main components: (1) teaching preparation and (2) teaching implementation, with each contributing 50% of the final grade. Under the teaching preparation category, we evaluate two key elements: the quality of the questions posed by students regarding their interestingness, insightfulness, and usefulness and the solutions regarding their correctness, appropriateness, and the variety of table or plot functions employed. For the teaching implementation category, we review the submitted teaching video to determine if the student can effectively explain concepts and demonstrate understanding. Three teaching assistants and instructors graded the assignment. Each assignment was graded by two graders, and any significant discrepancies in evaluations were discussed and resolved. In addition to the quantitative scores assigned for each category, we also give students qualitative feedback. This includes highlighting specific segments of the teaching video where the student explained a concept well, demonstrated clear evidence of learning, or revealed misconceptions.

5 STUDENTS' REFLECTION

We analyzed the students' responses to the open-ended question about their teaching experience: "How might the teaching experience help you become a better learner?" From the analysis, several themes emerged: (1) Teaching encourages retaining material through review, practice, and reinforcement; (2) Students discovered new methods for solving problems; (3) Many gained increased confidence in their data science and/or communication skills; (4) Teaching appears to enhance metacognition. As one student pointed out, *"The process of making a lesson plan could be applied to making a learning plan. It is the same. Identify what you want to learn, find what things you would most need to know, and learn them."* (5) Teaching motivates active learning. One student noted, *"This teaching experience helped me to see the perspective of the teachers. So, I can understand how frustrated they can get, especially if the student is not paying attention or that they are not even trying to participate. So, I think I can be a better learner by attempting to participate more often and asking questions. And try to pay attention better."*

6 DISCUSSION

From this pilot study, we have gathered preliminary evidence supporting the effectiveness of this assessment method. Students' reflections indicate that this approach potentially enhances their technical skills and fosters metacognition, self-efficacy, and communication. However, we have identified several challenges. For instance, considerations must be made for students for whom English is a second language (ESL) and those who may be naturally shy. The teaching performance may depend on the background of the tutees, which are hard to control for. Additionally, we recognized potential difficulties in grading when scaling up this approach. These observations motivate the need for future work to develop more equitable and efficient assessment strategies for complex skill sets as those we attempt to evaluate.

REFERENCES

[1] David Duran. 2017. Learning-by-teaching. Evidence and implications as a pedagogical mechanism. *Innovations in Education and Teaching International* 54, 5 (2017), 476–484.
[2] Rachel S Lim, Joe Gibbs Politz, and Mia Minnes. 2023. Stream Your Exam to the Course Staff: Asynchronous Assessment via Student-Recorded Code Trace Videos. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 144–150.

Pair Programming with ChatGPT

Xi Chen
xi.chen@uvu.edu
Utah Valley University
Orem, UT, USA

Jingsai Liang
jliang@westminsteru.edu
Westminster University
Salt Lake City, UT, USA

ABSTRACT

This poster explores the potential of ChatGPT to replace the traditional approach of pair programming in introductory computer science courses. Traditionally, two students collaborate as a driver and a navigator, periodically switching roles. Now, a student can pair up with ChatGPT, which offers an innovative approach to pair programming. This exploratory activity, which emphasizes collaboration and communication, provides step-by-step instructions for effectively interacting with ChatGPT during pair programming.

This poster reflects on the advantages and limitations of using ChatGPT in pair programming. The main advantages of using ChatGPT include rapid responses, syntax error-free code generation, and flexibility in handling incomplete pseudocode. The primary limitations include the coding generation style, redundancy in responses, and challenges in understanding the code. Despite the advantages, it may still be valuable to have students work with human partners in certain situations, particularly for learning purposes.

This poster proposes that ChatGPT is an invaluable tool for enhancing productivity and emphasizes the importance of becoming proficient in its use during students' college years. It also provides insights into the effective utilization of ChatGPT in pair programming and its preparation for future careers in programming and related fields.

ACM Reference Format:
Xi Chen and Jingsai Liang. 2024. Pair Programming with ChatGPT. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635600

1 INTRODUCTION

Pair programming [2] is a software development practice commonly used in introductory computer science courses , where two students collaborate in two roles: a *driver* and a *navigator*. Students periodically switch roles, and here are the tasks for each role:

- **Driver**: implements ideas, writes code, and assists in testing and debugging.
- **Navigator**: provides instructions, reviews, tests, and debugs the code.

ChatGPT and other LLM tools have garnered significant attention in the field of computer science education [1]. With this revolutionary phenomenon, ChatGPT has the potential to replace the traditional approach to pair programming by pairing one student with ChatGPT as a team. In this innovative approach, the student and ChatGPT take on roles as either the driver or navigator periodically.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635600

2 DESIGN

In this exploratory activity designed in an upper-level algorithms course, we provided step-by-step instructions and a demo on effectively interacting with ChatGPT 3.5 during pair programming using Python, and then assigned students to work on a short programming problem while paired with ChatGPT.

2.1 Instructions

To effectively interact with ChatGPT as a driver or navigator, we need to follow specific instructions. For instance, here are instructions for when a student is the navigator, and ChatGPT is the driver.

(1) You can start by setting up the role as a navigator and tell ChatGPT like this:

    We are software engineers. You and I will be pair programming on a project. We will use the driver-navigator model.

    I will be the navigator first. As a navigator, I'll provide the pseudocode and suggestions, and you will implement it in Python code. Do you understand?

(2) You can begin by describing the problem and providing the initial pseudocode. While the code generated by ChatGPT may seem perfect, it often fails to run correctly because it cannot consider for all corner cases and heavily relies on the pseudocode you provide, even if it is wrong.
    You don't have to provide a complete and error-free pseudocode to get started on coding. You can offer a draft and collaborate with ChatGPT to develop the pseudocode.

(3) You'll need to review the code by copying it and running it on your computer. If you encounter an error, you can ask ChatGPT for assistance or try to resolve it yourself. Afterward, you can engage with ChatGPT again to fix the bug.

(4) After confirming there are no obvious bugs in the code, you will need to test it. You can ask ChatGPT to provide the test cases, and then verify them both manually or with ChatGPT's assistance.
    You can also use ChatGPT to print out the internal results of a given test case and perform calculations during testing. It's advisable to avoid any manual work that ChatGPT can handle for you during the testing.

(5) Repeat steps 2 to 4 until a solution is reached. It typically takes dozens of rounds of conversations.

Providing high-quality prompts is essential for effective interaction with ChatGPT. The most effective prompt is one that is as

complete and clear as possible, where you specify the line number of the code, variable, formula, or step to reduce random and unhelpful responses.

However, there are some drawbacks that we caution students about. ChatGPT may not be as flexible or intelligent as we expect, often requiring our patience and guidance. For instance, ChatGPT may repeat the same mistakes even after correction, and when provided with incorrect instructions, it will follow those instructions without argument.

Additionally, we have shared some tips for working with ChatGPT. For example, if the code is too lengthy for ChatGPT to display at once, students can type *continue* to allow ChatGPT to continue the output. If the code is scattered throughout the conversation, students can ask ChatGPT to print the entire code without omissions. In cases where ChatGPT forgets its role, such as writing code while being a navigator, a reminder may be necessary to keep it on track.

2.2 Demonstration and Assignment

In the demo, we shared a complete chat history of solving the classic convex hull problem using the divide and conquer. The goal is to find the convex hull of a set of points in a 2D plane, which is the smallest convex polygon that encloses all the points. Students can review the demo to gain insight into the hands-on experiences in effectively serving as either the driver or navigator during pair programming.

Following the demo, students were assigned to work on a problem below, initially pairing with ChatGPT as the navigator.:

The input is an array A with n numbers. A pair of numbers from A with index i and j is said to be an inversion if i<j, but A[i]>A[j]. The output is the number of inversions in A. The goal is to develop an O(nlogn) solution.

3 REFLECTION

We summarized our reflections on this activity in the following aspects:

(1) **The benefits or advantages when pair programming with ChatGPT as compared to a human partner.**

We discovered numerous advantages of working with ChatGPT over a human partner. The foremost benefit is ChatGPT's ability to provide rapid responses without the need for waiting, extensive discussions, or interpersonal interaction, ultimately saving the problem-solving time. Importantly, unlike humans who are prone to making simple syntax errors and typos, ChatGPT consistently delivers code free from low-level errors, such as missing parenthesise, thus eliminating the need for potential debugging. The code generated by ChatGPT is consistently clean and well-organized.

We were impressed by the remarkable feature that ChatGPT doesn't require a fully correct pseudocode to begin working. It can understand incomplete pseudocode and produce code with minimal initial guidance, which speeds up the design stage.

We also noted that partnering with ChatGPT significantly reduces students' stress during pair programming since it's a machine. Unlike humans, there's no need to keep up with information exchange, meet partner expectations, and thus eliminate the stress of feeling left behind if one student is unable to keep up the pace.

(2) **The limitations or challenges when pair programming with ChatGPT as compared to a human partner.**

We recognized numerous limitations when using ChatGPT compared to working with a human. For example, the way ChatGPT generates code differs significantly from how humans write it. Humans typically write code step by step, logically building the solution line by line and chunk by chunk, whereas ChatGPT generates the entire code at once. This can make it challenging for humans to follow and understand the code, as it is hard to break down and difficult to debug the errors.

Furthermore, ChatGPT's responses often exhibit redundancy and repetition, failing to align with human problem-solving logic. Retrieving prior conversations can be challenging unless it is specified exactly where the dialogues and lines of code are.

Thus, most students may learn coding better if working with humans, even if it may be slower to complete the coding. Collaborating with humans may also lead to brainstorming creative ideas that machines find challenging, especially in larger and more complex projects.

(3) **Preference of pair programming with ChatGPT or human partner.**

While we acknowledge the effectiveness of using ChatGPT to enhance productivity, we still appreciate the value of working with a human partner, particularly in class. The primary reason is that ChatGPT excels at providing solutions to simple and small-scale questions typically encountered in the class. When relying solely on ChatGPT, the opportunity for meaningful learning as students is limited. In the class, ChatGPT undoubtedly serves as a valuable tool for tasks like debugging, providing quick answers, and improving and commenting the code. However, in most cases, collaborating with a human partner in the class is more beneficial in learning.

(4) **Preparation for the future career in programming together with ChatGPT.**

If the students have never used ChatGPT before, this activity undoubtedly added a new tool to their skill set. Even for the students who have prior experience with ChatGPT, this activity offered a great opportunity for them to learn the most effective ways to interact with ChatGPT. Throughout this activity, it became evident that ChatGPT is an invaluable tool for enhancing productivity, emphasizing the importance of gaining proficiency during their time in college.

4 CONCLUSION

Overall, this study allowed students to explore the most effective ways to interact with ChatGPT, showcased its potential impact on learning code, provided students with valuable experience in working with an AI assistant, and prepared students for future careers.

REFERENCES

[1] Jenny T. Liang, Chenyang Yang, and Brad A. Myers. 2023. A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges. arXiv:2303.17125

[2] Charlie McDowell, Brian Hanks, and Linda Werner. 2003. Experimenting with Pair Programming in the Classroom. In *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education* (Thessaloniki, Greece) *(ITiCSE '03)*. 60–64.

Advancing Equity through a Research-Practice Partnership: A Case Study of a Professional Development Initiative for Educators of Color

Joshua Childs†
Educational Leadership and Policy
University of Texas at Austin
Austin, TX USA
joshuachilds@utexas.edu

Kaitlin Ogden
Education Leadership and Policy
University of Texas at Austin
Austin, TX USA
kait.ogden@utexas.edu

Carol Fletcher
Texas Advanced Computing Center
Austin, TX USA
cfletcher@tacc.utexas.edu

Allen Antoine
Expanding Pathways in Computing
Texas Advanced Computing Center
Austin, TX USA
aantoine@tacc.utexas.edu

Stephanie Baker
Texas Advanced Computing Center
University of Texas at Austin
Austin, TX USA
snbaker@tacc.utexas.edu

Alejandro Madrigal
University of Texas at Austin
Austin, TX USA
traemadrigal@utexas.edu

ABSTRACT

There is growing consensus that focused attention on increasing the computer science (CS) certification of Black and Brown teachers is needed. More certified CS Black and Brown teachers can propel equity-based CS principles, support diverse CS classrooms, and diffuse culturally relevant teaching practices. The Computing Educator Diversity Initiative (CEDI) is a tailored in-person and online professional development program that supports the growth and development of CS knowledge and inclusive pedagogy for educators. CEDI specifically recruits teachers who reflect the racial and ethnic identities of the student populations in their schools. Through a Research-Practice Partnership (RPP) that facilitated use of improvement science tools, culturally relevant professional development, and collaboration, CEDI assisted aspiring CS teachers to become certified. Findings from two cohorts of CS teachers revealed that engaging within a cohesive community of practice led to the cultivation of CS expertise, pedagogical proficiencies, and equitable teaching methodologies.

CCS CONCEPTS

•Social and professional topics~Professional topics~Computing education~K-12 education

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635633

KEYWORDS

equity; professional development; community of practice

ACM Reference format:

Joshua Childs, Kaitlin Ogden, Carol Fletcher, Allen Antoine, Stephanie Baker, and Alejandro Madrigal. 2024. Advancing Equity through a Research-Practice Partnership: A Case Study of a Professional Development Initiative for Educators of Color. In *Proceedings of 55th ACM Technical Symposium on Computer Science Education V.2* (SIGCSE 2024), *March 20-23, 2024, Portland,* OR. ACM, New York, NY, USA.

1 INTRODUCTION

There has been growing consensus around computer science education as a fundamental component for preparing students in an increasingly technology-driven world. Although computer science (CS) courses have been traditionally offered as elective courses or through tracked course pathways, research has highlighted how policymakers, researchers, parents, and employers are increasingly recognizing the importance of (CS) education for all students. This trend has accelerated since the COVID-19 pandemic, with many businesses, social sectors, and educational institutions relying more frequently on technology to exchange information, collaborate, and communicate efficiently. Arguments exist that coding is the new literacy that should guide policy decisions at the national, state, and local level [1]. Although improvements in access and participation have increased, traditionally underserved populations in computing (e.g., women, students of color) remain minimally represented in K-12 computing, college computing, and in the computing workforce [2].

CS has evolved from being an enrichment subject for specific groups of students to many states making CS a core component of their educational curriculum [3]. Numerous resources have targeted Black and Latine students to increase course enrollment

and completion [4]. However, student-level investments should coincide with increasing the number and capacity CS teachers. Furthermore, within K-12 CS education there is a critical need for more Black and Latine CS teachers who are more likely to lead classrooms in schools that represent similar racial demographics.

2 TEACHER REPRESENTATION IN CS

Nationally, educators of color represent 20% of the K-12 teaching workforce but leave teaching at rates 25% higher than their white counterparts [5]. *The Computer Science Teacher Landscape: Results of a Nationwide Teacher Survey* reported disconcerting statistics about the CS teaching force in the United States, such as that nearly 75% of CS teachers are White and most CS teachers work in schools that are affluent and not racially diverse. Similar patterns related to the race/ethnicity of CS teachers have been found in Texas. Disparities exist in the racial/ethnic composition of the student population as compared to the overall teacher workforce and the CS teacher workforce. For example, White students make up 27% of the Texas student population in 2020-21 but 64% of the TX CS teacher population. The mismatch between teachers and students is most pronounced for Latine students, who make up 53% of the student population but only 20% of the CS teacher population. Disparities also exist for Black students as compared to Black CS teachers, but on a smaller scale. In Texas, the underrepresentation of Black and Latine certified CS teachers may contribute to the equity gaps in enrollment between White students and their minoritized peers. The Computing Educator Diversity Initiative (CEDI) seek**s** to support the recruitment and retention of CS teachers who reflect their students' racial and ethnic identities through a minoritized teacher affinity group. The effectiveness of racial affinity groups in nurturing teachers of color has been well documented as a tool for supporting teacher retention and for helping teachers to navigate the challenges of addressing equity in their classrooms and schools [6].

3 EVALUATION PROCEDURE

This study implemented a case study design within the larger CEDI project. Teachers represented 15 different districts across the state of Texas and worked in school districts that are in rural, urban, and suburban settings. Before the first CEDI professional development (PD) session, each CEDI teacher completed a 20-item survey that asked questions related to their capacity, empowerment, belonging, and confidence related to learning and teaching computer science. Additionally, 6 CEDI PD instructors completed a 10-item survey related to their PD practices and support effort within the initiative. Also, we interviewed each CEDI participant and instructor at the end of the CEDI PD process. Our initial findings suggest that CEDI helped prepare teachers for the CS certification exams and created a community among participants for support and learning opportunities.

4 CONCLUSION

Our preliminary findings found that a community of practice and a quality PD program can help prepare and support aspiring CS teachers. Our findings appear to suggest that aspiring CS teachers feel supported when instructors not only provide relevant content knowledge but also offer support and resources that can assist in their teaching practices. Our poster highlights the need for understanding the PD experiences of teachers as they learn to become more confident in their CS teaching practices and understanding of CS curricula. PD programs serving CS teachers of color are important and are useful for supporting the success of minoritized teachers.

.

REFERENCES

[1] New Media Consortium. 2017. *NMC/CoSN Horizon Report: 2017 K-12 Edition.*

[2] Code.org, CSTA, and ECEP Alliance, 2021. *State of Computer Science Education: Accelerating Action through Advocacy.* Seattle, Washington: Code.org. https://advocacy.code.org/stateofcs

[3] Beryl Nelson, 2014. The data on diversity. Communications of the ACM, 57(11):86-95.

[4] National Science Foundation, 2012. *Computer and Information Science and Engineering (CISE) Strategic Plan for Broadening Participation*. Alexandria, VA: National Science Foundation.

[5] Nicole D. Martin, Stephanie N. Baker, Madeline Haynes, and Jayce R. Warner. 2023. The motivation to teach computer science (mtcs) scale: Development, validation, and implications for use. *Computer Science Education* (2023),1–18.

[6] Aman Yadav and Marc Berges. 2019. Computer science pedagogical content knowledge. ACM Transactions on Computing Education 19, 3 (2019), 1–24.

Data Science Mastery Learning Using Parsons Problems-Inspired Table Transformations

Jacob Seungwon Choe
ownssi@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Matthew G. Lee
matthewlee3@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Siddharth A. Marathe
smarathe@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Armando Fox
fox@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Dan Garcia
ddgarcia@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Narges Norouzi
norouzi@berkeley.edu
UC Berkeley
Berkeley, CA, USA

ABSTRACT

Table transformations are a critical skill to master in order to fluently work with data. In introductory data science courses, however, students have found these transformations particularly challenging to learn. One complex transformation is the pivot transformation, which reorganizes a table based on aggregation and summarizing along selected columns and rows. Current assessments test student understanding in static scenarios. Thus, there is an opportunity to help students explicitly work through the steps and variables needed to express a pivot transformation in a randomizable manner. As such, we explore whether a dynamic digital assessment for the pivot transformation can effectively achieve mastery learning towards this skill. Our design is inspired by Parsons problems, in which answer components (pivot table output labels and values) can be composed into the output of a pivot transformation. A question can be derived from a small randomized dataset, and randomized `Pandas` code that operates upon the dataset and can be autograded. We plan to conduct a pilot study with data science students to investigate whether 1) using a programmable online platform to practice pivot tables helps improve performance on exams, and 2) randomization and instant feedback on the online platform contribute to improved student learning.

ACM Reference Format:
Jacob Seungwon Choe, Matthew G. Lee, Siddharth A. Marathe, Armando Fox, Dan Garcia, and Narges Norouzi. 2024. Data Science Mastery Learning Using Parsons Problems-Inspired Table Transformations. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635556

1 INTRODUCTION

Demand for data scientists has increased in recent years, resulting in growth in data science curricula offered in K-12 and higher education [2]. Tabular transformations, the manipulation of one table of data to another table of data, are an important category of skills to master for any data science student, yet instructors often find tabular transformations difficult to teach due to the complex steps involved in the transformation algorithms [1, 5]. Through our experience in teaching an introductory data science course, we have found similar perspectives. In particular, the pivot transformation, a transformation where the data in the original table is grouped by one category along the rows and another category along the columns, is a particularly tricky concept for students to learn due to the large number of potential variables that can influence the results such as data fields and aggregation functions.

Mastery learning is an educational paradigm that has redefined how students should be held accountable for their learning as they progress through their education [3]. There has been exploratory work in bringing this framework into data science education [6]. Furthermore, randomized assessments with instant feedback are critical for implementing the self-paced nature and mastery-based progression of mastery learning, a concern of increasing relevance as data science courses continue to scale with student demand [4]. Current static digital approaches for assessing student understanding of tabular transformations like Jupyter Notebooks lack these features; hence, we see an opportunity to explore whether mastery learning can help enhance student proficiency of pivot transformations. Questions utilizing graphics, like Parsons problems, can help students learn programming concepts by providing scaffolding to build toward expert solutions [7]. We anticipate that this paradigm can be extended to pivot table transformations and ultimately help operationalize mastery learning in the data science curriculum.

2 METHODOLOGY

Students often have difficulty starting the assignment in the current format, in which they must simply produce code that generates a given pivot table. To address this, we built a reverse-engineered formative assessment for students to practice this skill. Our new problem framework will be on an interactive online platform, visualized in Figure 1. Students will be given a randomized question that includes an original data table and a pivot transformation expressed as `Pandas` code. Students will then be given a bank of column, index, and data pieces; some of these pieces are correct and others will be distractors (e.g. incorrect aggregation functions, incorrect labels). They can drag these pieces into pre-populated answer slots to build the correct pivot table output.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635556

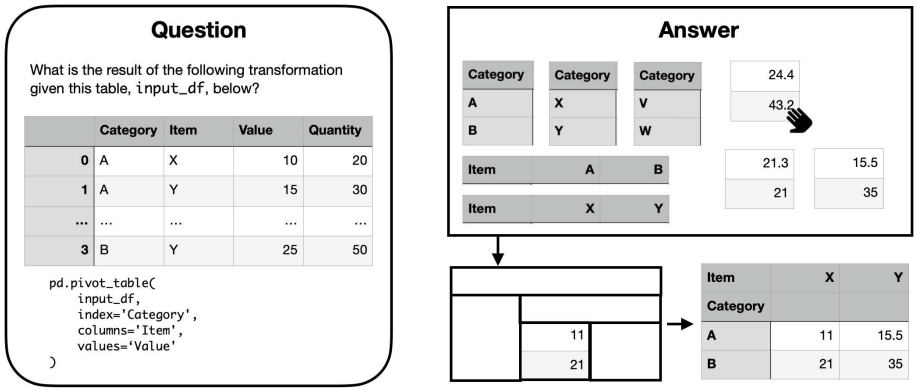


Figure 1: A visualization of our digital assessment (with randomization and instant-feedback capabilities) question with the original data table and the corresponding transformation is provided. The student must drag and drop table pieces to fill in columns and labels while ignoring distractors.

We hypothesize that this new style of question generation and assessment in data science, with randomization and instant feedback features, will improve students' mastery of the topic and improve their learning efficiency. Specifically, our research questions are:

RQ1 Does using a programmable online platform to practice pivot tables help improve performance on exams compared to traditional static assignments in data science courses?

RQ2 Does randomization and instant feedback on the online platform contribute to improved student learning?

Using our prototype, we will explore how an online platform to test critical data science concepts like pivot transformations will help to diagnose common student misconceptions. We believe the randomization features and instant feedback available on the online platform will contribute to better performance on subsequent examinations since it enables students to attempt different versions of the problem and prevents students from retaining incorrect mental models of how concepts work for future problems.

3 FUTURE WORK

To understand if assessing pivot transformations on the online platform works as hypothesized, we plan to conduct an experimental study. We will recruit study participants from a large introductory data science course, consisting of students with little to no experience in table transformations, and randomly split the students into two groups: one that does a baseline Jupyter Notebook assignment and the other that takes the assignment on the online platform. Both modalities would include similar assessments. As a primer to the assignment, students will be able to watch a short video describing the concept of pivot tables. For those taking the Jupyter assignment, we will grade their answers after the session is completed and return their scores. The online platform users will be provided with instant feedback after every question they attempt on the assignment. We would collect feedback from these interviews to measure the impact of our experimental modality in reference to the current baseline format.

4 CONCLUSION

We present a new approach for assessing tabular transformations, a key data science principle. Randomization and instant feedback are incorporated as features in our exercises, which will be evaluated in a pilot study to examine the effectiveness of our platform.

REFERENCES

[1] Ridha Alkhabaz, Zepei Li, Sophia Yang, and Abdussalam Alawini. 2023. Student's Learning Challenges with Relational, Document, and Graph Query Languages. In *Proceedings of the 2nd International Workshop on Data Systems Education: Bridging Education Practice with Education Research* (Seattle, WA, USA) *(DataEd '23)*. Association for Computing Machinery, New York, NY, USA, 30–36. https://doi.org/10.1145/3596673.3596976

[2] Francine Berman, Rob Rutenbar, Brent Hailpern, Henrik Christensen, Susan Davidson, Deborah Estrin, Michael Franklin, Margaret Martonosi, Padma Raghavan, Victoria Stodden, and Alexander S. Szalay. 2018. Realizing the Potential of Data Science. *Commun. ACM* 61, 4 (mar 2018), 67–72. https://doi.org/10.1145/3188721

[3] Dan Garcia, Armando Fox, Solomon Russell, Edwin Ambrosio, Neal Terrell, Mariana Silva, Matthew West, Craig Zilles, and Fuzail Shakir. 2023. A's for All (As Time and Interest Allow). In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1042–1048. https://doi.org/10.1145/3545945.3569847

[4] Sam Lau, Justin Eldridge, Shannon Ellis, Aaron Fraenkel, Marina Langlois, Suraj Rampure, Janine Tiefenbruck, and Philip J. Guo. 2022. The Challenges of Evolving Technical Courses at Scale: Four Case Studies of Updating Large Data Science Courses. In *Proceedings of the Ninth ACM Conference on Learning @ Scale* (New York City, NY, USA) *(L@S '22)*. Association for Computing Machinery, New York, NY, USA, 201–211. https://doi.org/10.1145/3491140.3528278

[5] Sam Lau, Sean Kross, Eugene Wu, and Philip J. Guo. 2023. Teaching Data Science by Visualizing Data Table Transformations: Pandas Tutor for Python, Tidy Data Tutor for R, and SQL Tutor. In *Proceedings of the 2nd International Workshop on Data Systems Education: Bridging Education Practice with Education Research* (Seattle, WA, USA) *(DataEd '23)*. Association for Computing Machinery, New York, NY, USA, 50–55. https://doi.org/10.1145/3596673.3596972

[6] Allison S. Theobold. 2023. Human Centered Data Science: Ungrading in an Introductory Data Science Course. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) *(ITiCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 327–333. https://doi.org/10.1145/3587102.3588816

[7] Nathaniel Weinman, Armando Fox, and Marti A. Hearst. 2021. Improving Instruction of Programming Patterns with Faded Parsons Problems. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 53, 4 pages. https://doi.org/10.1145/3411764.3445228

Dynamic, Animated Feedback for Randomized Problems with Computer-Based Testing

Arihant Choudhary
arihant@berkeley.edu
University of California, Berkeley
USA

Dan Garcia
ddgarcia@berkeley.edu
University of California, Berkeley
USA

ABSTRACT

Computer-based testing (CBT) is becoming more common in CS Education spaces, and offers many advantages over paper-based modalities. For example, computerized assessments make it easier for students to learn from a wider range of questions in problem sets through problem randomization. Students can then work on problem after problem in a formative assessment mode of learning. While computer-based assessments help facilitate randomized problem *generation*, solution *feedback* is typically delivered in static text or image formats, or perhaps a video may be provided. Typically these show or describe the instructor solving a *generic* problem, not *this particular* randomized problem.

The Dynamic Animation Feedback Tool (DAFT) aims to support mastery learning by helping instructors automatically create algorithm visualization animations to explain their solutions, *customized to the problem the student was given.* The instructor would write code to randomly generate a graph using the CBT system, and ask the student some algorithmic question (e.g., what's the sequence of nodes visited in a depth-first search). Then, *that same graph* would have its nodes animated (by running the hidden instructor solution to "fill in" each node as it was visited). This allows an instructor to ask a question about *any* algorithm on *any* randomized graph, and if the student didn't get it right, the animation of the solution on that *exact same graph* would be available. DAFT provides a visually rich, problem-specific animated solution that hopes to be valuable resource for students to interpret problem solutions and achieve mastery learning.

KEYWORDS

algorithm visualization, animated feedback, mastery learning

ACM Reference Format:
Arihant Choudhary and Dan Garcia. 2024. Dynamic, Animated Feedback for Randomized Problems with Computer-Based Testing. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635623

1 BACKGROUND

Computer-based testing (CBT) is being used to promote a positive testing culture in schools, in which assessments are viewed as a beneficial tool for learning rather than just a grading toolkit. It offers advantages over its paper-based modalities and has been recognized as a progressive instructional tool for both educators and learners [3]. For example, CBT increases the combinatorial prospects of creating tests for instructors and makes it easier for them to create a wider range of questions in problem sets through problem randomization.

Computer Science educators often employ visualizations to provide learners with another way to look at a particular problem, algorithm, or data structure. This approach has been shown to improve engagement among students in computer science [1, 2]. It is particular useful when explaining dynamic concepts such as graph traversals, search algorithms, sorting, etc. Static text or images are often not enough to explain the actual *flow* of the algorithm – one sometimes needs to see an *animation* to fully capture the idea.

However, acomputer-based testing (CBT) platforms often do not use animation tools to provide feedback for these randomized problems. They will typically link to a video of the instructor solving a similar problem, assuming the student can get that idea from that, and map it over to the one they are working on.

In the poster, we wish to introduce a tool that can help deliver animated feedback to randomized problems in CBT environments: (a) to provide custom feedback to the specific problem the student was provided, and (b) to offer visually enriched, solution-specific feedback, for a more interactive and effective learning environment.

2 DYNAMIC ANIMATION FEEDBACK TOOL

The Dynamic Animation Feedback Tool (DAFT) addresses the lacuna in current computer-based testing environments by providing an adaptive (to exactly the problem at hand) and animated solution. The instructor authors python code to generate the randomized graph, which is sent to *GraphViz* to render and display on the question, along with the question prompt. For example, *What is the order of the nodes that will be visited by a Depth First Search algorithm?*

The instructor then calls their internal hidden solution code on the graph, and generates "frames" for each important step in the animation, by calling *GraphViz* with some sets of nodes highlighted with a fill color of red, say. Finally, the instructor passes these frames to DAFT, which auto-generates an animation, and presents it to the user if they get the question incorrect. We also plan to provide a simple video-style interface, with "rewind", "play", "step", and "stop" buttons. Figure 1 shows various frames of this step-by-step execution of the Depth-First-Search (DFS) algorithm on a random tree. This isn't merely a visualization, but an instructional animation that breaks down the problem's solution into digestible, sequential steps, maintaining synchrony with the learner's problem instance.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635623

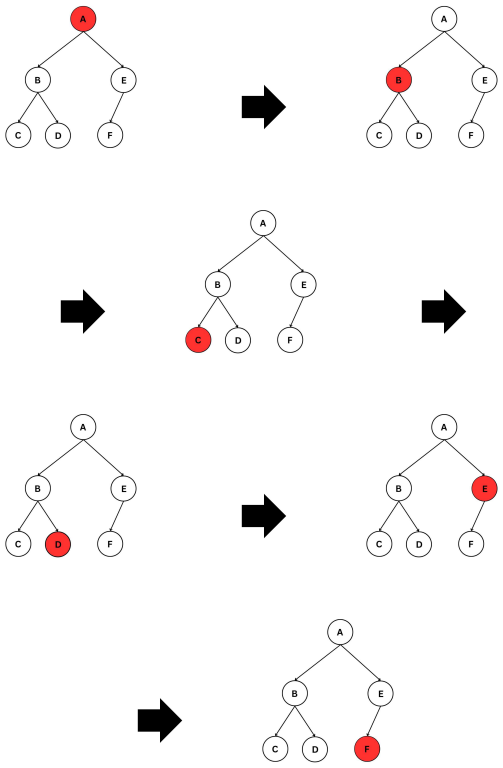


Figure 1: Sequences of the Animated Feedback Tool explaining the step-by-step execution of the Depth-First-Search (DFS) algorithm on a random tree.

3 FUTURE WORK

Although DAFT is a currently a work in progress, we are excited about several future directions for this endeavor.

Empirical Validation of DAFT: We plan on conducting a study to measure and analyze the impacts of animated feedback, contrasting its efficacy against generalized prerecorded solution videos and the absence of any feedback. We will assess metrics such as learner engagement, concept mastery, and student performance to gauge the practical impact of DAFT in the CBT settings.

Gen-AI Powered Custom Lecture Generation: Utilizing advancements in generative AI technologies, future iterations of DAFT could harness the capability to create custom feedback lectures. By synthesizing the voices of course instructors and using generative AI-driven animations, DAFT can generate personalized, informative, and instructor-specific lectures or feedback sessions that elaborate on solutions to randomized problems. This ensures continuity in instructional delivery and maintains a personalized touch in the feedback mechanism.

Inclusion of More Algorithm and Problem Types: Broadening DAFT's scope to encompass a wider array of algorithms and dynamic problems beyond just graph traversals is crucial. Future adaptations will incorporate mechanisms to visualize and explain diverse algorithmic domains and problem types, ensuring that DAFT is versatile and applicable across a myriad of topics within CS education.

Learner Experience and Accessibility: Ensuring that DAFT is not only instructive but also intuitive and accessible to all learners will be vital. Future enhancements will focus on optimizing the user interface, user experience, and ensuring accessibility, thereby making sure that the tool is inclusive and facilitates a universally accessible learning environment.

REFERENCES

[1] Eric Fouh, Monika Akbar, and Clifford A. Shaffer. 2012. The role of visualization in Computer Science Education. *Computers in the Schools* 29, 1–2 (2012), 95–117. https://doi.org/10.1080/07380569.2012.651422

[2] Thomas L. Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, and J. Ángel Velázquez-Iturbide. 2002. Exploring the Role of Visualization and Engagement in Computer Science Education. (2002), 131–152. https://doi.org/10.1145/960568.782998

[3] Martin J. Tomasik, Stéphanie Berger, and Urs Moser. 2018. On the Development of a Computer-Based Tool for Formative Student Assessment: Epistemological, Methodological, and Practical Issues. *Frontiers in Psychology* 9 (2018). https://doi.org/10.3389/fpsyg.2018.02245

Champion, Collaborator, Curator, Anchor: Supporting K-12 Teachers to Integrate Computational Thinking

Merijke Coenraad
Learning Experience Design
Digital Promise Global
Washington, D.C., USA
mcoenraad@digitalpromise.org

Alessandra Rangel
Learning Experience Design
Digital Promise Global
Redwood City, CA, USA
aramgel@digitalpromise.org

Quinn Burke
Learning Experience Design
Digital Promise Global
Redwood City, CA, USA
qburke@digitalpromise.org

Kelly Mills
Learning Experience Design
Digital Promise Global
Washington, D.C., USA
kmills@digitalpromise.org

Pati Ruiz
Learning Sciences Research
Digital Promise Global
Redwood City, CA, USA
pruiz@digitalpromise.org

Kyle Dunbar
Learning Experience Design
Digital Promise Global
Washington, D.C., USA
kdunbar@digitalpromise.org

ABSTRACT

Computational thinking (CT) provides opportunities for students to build the skills they need for everyday life, future academic success, and careers. Increasingly, districts are integrating CT into K-12 disciplinary subjects and a growing number of teachers need the knowledge to lead CT learning opportunities. CT is still nascent in teacher education, however, and not all teachers are receiving professional learning on the job. Teachers need effective methods of support to ensure they are providing students with high quality CT learning opportunities. We present methods of support for integrating CT into schools. The poster highlights four methods to support teacher professional learning: allowing ample time for collaboration, a school-based advocate championing CT integration, posting and promoting curated resources, and a foundational person anchoring the work as the initiative expands. In doing so, this poster provides interrelated examples of teacher support and highlights methods of continuous support for teachers integrating CT across content areas.

ACM Reference Format:

Merijke Coenraad, Alessandra Rangel, Quinn Burke, Kelly Mills, Pati Ruiz, and Kyle Dunbar. 2024. Champion, Collaborator, Curator, Anchor: Supporting K-12 Teachers to Integrate Computational Thinking. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635498.

1 INTRODUCTION

CT has been recognized as an essential problem-solving skill and is increasingly a requirement within schools and districts [1,2]. Concurrently, focus has shifted toward CT integration within content areas [3] rather than stand-alone courses. This expansion has resulted in the need for a greater number of teachers of CT, including teachers never formally trained in computing. As CT expands, there is a need to effectively support teachers to ensure students are receiving high quality CT learning opportunities.

We present methods of support for the integration of CT through case studies from four school districts. We answer the research question: *What district-supported professional learning features or systems support teachers to integrate computational thinking?* This poster provides examples of methods of teacher support and adds to the growing body of work surrounding teacher professional learning (PL) and support by expanding conversations beyond PL workshops to consider on-going, personalized ecosystems of support.

We completed a landscape study to better understand the current scope and state of CT within our partner districts. This study used a mixed-methods design wherein we first developed the survey in partnership with our practice-side district partners and used findings to develop and implement focus group protocols to further explore the quantitative results. We analyzed multiple choice and Likert-scale questions using descriptive and count statistics and used a two-round qualitative coding process to analyze open-response survey questions. Focus group data was coded through an inductive thematic analysis process.

2 FINDINGS

In the study, teacher responses showed a need to go beyond traditional PL models and specified supports that would better enable them to integrate CT. Teachers highlighted four areas beyond generalized PL: teaching resources, access to a dedicated teacher or point person to provide coaching and one-on-one support, collaboration opportunities with other teachers, and the opportunity for students to have a specific stand-alone course on CT/CS. We present four case studies describing the unique approaches that each district applied to support CT integration in alignment with these areas of need: the guidance of a champion advocating for the work, time for collaboration with peers, curated

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20-23, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03 https://doi.org/10.1145/3626253.3635498

resources, and a person anchoring the work as the initiative expands.

2.1 Champion: A school-level point person

A central method of support in Talladega County Schools (TCBOE) is having **champions of CT integration** within each school. These champions, an Expert Down the Hall and Dynamic Learning Partner, are point-people that advocate and lead the effort of CT integration within the school building. The value of these advocates is represented by survey and focus group data. When discussing current support for teaching or integrating CT/CS, over three-quarters of teachers surveyed agreed or strongly agreed they were supported by administrators and fellow teachers (78.57%; 11 of 14) and they had people to talk to when they had questions (85.71%; 12 of 14). All of the teacher respondents (14) in TCBOE reported they were supported to integrate CT/CS in their classroom by teachers they worked with regularly and 92.86% of teachers (13 of 14) reported they have been supported by content experts such as coaches or mentors. When asked about resources for integrating CT, one teacher noted, "I look at the state standards for suggested lessons and talk with our Tech Coach."

2.2 Collaboration: Time to work with peers

North Salem Central School District has developed a district-wide vision for their CT Pathway motivated by their district mission and aligned to their Profile of a Graduate. Currently, their district focus is on capacity building through curriculum development, resource acquisition, and PL. Their context as a small district with two schools (one elementary and one secondary) allows PL to occur across grade levels and content areas. In North Salem, teachers can turn to each other as **partners and collaborators**. The majority of teachers in North Salem agreed that they had people to talk to when they had questions about CT/CS (76.24%; 77 of 101) and teachers they regularly work with supported them in integrating CT/CS in their classroom (81.19%; 82 of 101). Teachers in North Salem reported they look to colleagues for additional resources and over half of teachers reported that they are well supported by administration and fellow teachers to teach or integrate CT (65.35%; 66 of 101).

2.3 Curated: A repository of vetted resources

In Broward County Public Schools (BCPS), the STEM+CS program, part of the Applied Learning department, **provides curated and vetted supports for teachers** and 25.70% of teachers (46 of 179) reported using these district resources, including Clever to access CT/CS apps (5.59%; 10 of 179), Canvas courses (4.47%; 8 of 179), the district webpage (1.68%; 3 of 179), and the STEM+CS program itself (3.35%; 6 of 179). Having a district program leading the effort means teachers and school leaders have a suite of resources reviewed by district officials, eliminating the barrier that teachers express around finding materials to integrate CT.

2.4 Anchor: Grounding CT/CS with an LMS

Kettle Moraine School District (KMSD) is currently growing their CT Pathway based on foundational work in the elementary grades. CS integration is required by the school district for all students K-5. KMSD is building upon the existing CS framework to embed CT skills through integration opportunities and collaboration between the library media specialist (LMS) and classroom teachers. In this support system, **the LMS is an anchor for the initiative**, using their flexible schedules to push into classrooms to teach CS skills and collaborating with teachers on integration opportunities and methods of embedding CT skills. Uniquely in this model, student knowledge is built in the LMS-led CT/CS learning experiences and then their CT knowledge is applied across disciplines and used to enhance discipline-specific learning experiences. The power of this support model is demonstrated in the teacher data. In KMSD, 83.67% of teachers (41 of 49) reported they were supported to integrate CT/CS by teachers they worked with regularly and 55.10% of teachers (27 of 49) reported they are supported by content experts. When discussing where they go when looking for resources, 34.78% of teachers (8 of 23) reported that they turn to CT/Tech experts, including their LMS. One teacher shared, "I'm sure I could look online, but our library media specialist has a ton of knowledge, and she's my go-to person."

These case studies demonstrate how districts go beyond PL focused solely on workshops. In future work, we plan to examine how these support systems work in tandem within single districts as we examine our broader district network. Having sustained professional learning with point people and wrap-around support in the form of a collaborator, champion, anchor, and curated resources show promise for creating broader networks of PL and ensuring that as CT expands within K-12 schools teachers have the dedicated support they need. While these methods of support are not necessarily novel, the case studies provide perspective on how these supports can be implemented for an entire system at the district level. These ecosystems of continuous support through a variety of methods are becoming increasingly important to ensure that students are receiving quality instruction and teachers have the support they need to be confident in what they are teaching.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2219350 & 2219351.

REFERENCES

[1] Code.org, CSTA, and ECEP Alliance. 2022. *2022 state of computer science education: Understanding our national imperative.* Retrieved from https://advocacy.code.org/stateofcs

[2] Katrina Falkner, Sue Sentance, Rebecca Vivian, Sarah Barksdale, Leonard Busuttil, Elizabeth Cole, Christine Liebe, Francesco Maiorana, Monica M. McGill, and Keith Quille. 2019. An international comparison of K-12 computer science education intended and enacted curricula. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research.* DOI: https://doi.org/10.1145/3364510.3364517

[3] Irene Lee and Joyce Malyn-Smith. 2020. Computational thinking integration patterns along the framework defining computational thinking from a disciplinary perspective. *J. Sci. Educ. Technol.* 29, (2020), 9–18. DOI: https://doi.org/https://doi.org/10.1007/s10956-019-09802-x

Student Preconceptions of Artificial Intelligence: Results from Single Institution Survey

Noah Q. Cowit
University of Colorado Boulder
Department of Information Science
Noah.Cowit@colorado.edu

Casey Fiesler
University of Colorado Boulder
Department of Information Science
Casey.Fiesler@colorado.edu

ABSTRACT

Artificial intelligence (AI) has become an increasingly critical component of not only the computing workforce but also society. It is essential for a diverse group of young people to contribute to this field. However, even within computing, AI is not taught to all post-secondary students. Students often must self-select into AI courses, meaning their reasons for choosing AI may be based on preconceptions of the discipline that may or may not be accurate. We extend the work of a small-n interview study of primarily Asian/Asian American undergraduate students, many of whom expressed perceptions of AI that paralleled identified computing stereotypes. Many of these stereotypes have the potential to discourage undergraduate computing students to take classes or specialize in AI, particularly those from underrepresented groups. Here we present a larger scale validation of those findings in the form of survey data conducted at a large public research institution in the USA. The survey largely confirmed the findings of the interview study at a larger scale, and we also found that gender did not significantly influence the results. Finally, we discuss strategies for AI integration into non-AI computing courses based on those previously used in responsible computing contexts, the goal being to counter harmful preconceptions before students specialize into computing subareas.

AI has already made a great impact on a variety of computing and non-computing related disciplines, and is poised to play an increasing role across various areas in industry and society [1, 6, 7, 12, 13]. It is essential to educate young people to contribute to this field to ensure the development of a high-qualified workforce. This requires post-secondary computing students to sign on to learn about the discipline. However, within university computing departments AI is not always a part of the required undergraduate or graduate curriculum, meaning computing students must choose whether to take courses and further their education in AI based on their already existing opinions on the subject.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635484

One recent SIGCSE paper, "Computing Specializations: Perceptions of AI and Cybersecurity among CS Students" used interview methods to identify a variety of preconceptions related to AI: that AI is very difficult and time consuming "intimidating" "rigorous"; AI requires advance meth skills "I think all of AI/ML is essentially just math."; AI is "trending" and "cool"; AI requires an inherent brilliance "they're really smart."; AI will have a large societal impact (although not always for the better); and AI is a "male-dominated" discipline [11]. Many of these preconceptions were noted as matching preconceptions of computing disciplines more generally [9] and potentially having a discouraging impact on marginalized or historically excluded groups in computing environments, particularly women [8]. In this poster, we aim to validate the findings of Ojha et al. with quantitative data from a single institution survey of post-secondary computing students in the USA. To this end, we ask the following research questions.

1.) To what degree are the preconceptions of AI identified in Ojha et al. (2023) confirmed by a larger sample of post-secondary computing students at a large public US university?
2.) To what extent are there difference in preconceptions of AI based on gender?[1]

ACMReferenceFormat:

Noah Q. Cowit, Casey Fiesler. 2024. Student Preconceptions of Artificial Intelligence: Results from Single Institution Survey. *In Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635484

1 METHODS

All data was collected from a single survey fielded during May and June 2023. In total, 106 responses were recorded from 24 information and computer science courses.

Sample Development: To gather our sample, we sent an email asking all instructors currently teaching computer science and information science courses at our university to field the survey to their students. Students were not offered an incentive to take the survey. The sample is not random, and students who had more positive opinions of their instructors may have been more likely to take the survey.

Major: Most respondents majored in computer (49.6%) or information science (42.1%). Small numbers were computational math/physics majors (3%) and engineering (5.3%) majors.

[1]Our sample size did not allow for comparisons based on other demographic features such as race or ethnicity.

Year of School: First year computing students made up 6.3% of our sample, Sophomores made up 11.3%, Juniors made up 26.3%, Seniors made up 31.3%, 5th/6th year undergraduates made up 10%, and graduate students made up 10%.

Gender: Men made up 67.1% of our sample, women made up 30.4%, and 2.5% preferred not to answer. No participants reported being non-binary or gender queer.

Race: White people made up 72.5% of our sample, South Asian/Indian people 10%, Hispanic people 7.5%, East Asian people 5%, Black people 5%, prefer not to answer 2.5%, American Indian people 1.3%, and 1.3% not listed.

Survey Items. One initial survey question asked students to choose the three subjects they thought were most and least important to becoming a professional in computing and information technology. Twelve subjects were presented to participants, adapted from the ACM computing classification system [3] with some modifications to improve accessibility and increase construct validity. The other eight survey questions were directly informed by the themes identified in Ojha et al. and were presented as four option Likert scales, with a "don't know" option to avoid midpoint ambiguity [10].

Analysis. Fisher's exact hypothesis tests were used to compare group differences between men and women. All other statistics were analyzed descriptively.

2 RESULTS AND DISCUSSION

In terms of topic importance, AI was considered a top three most important computing topic by 28.2% of participants, with only 14.8% of participants reporting it a top three least important topic out of the twelve topics asked about. Most participants (57%) did not consider AI a top three most or least important computing topic.

Table 1: Student Preconceptions of AI

Survey Item	N	$\bar{x}$	σ	p-value (gender)
Work in AI is difficult.	106	3.5	.67	.08
Work in AI is time-consuming.	106	3.6	.53	.60
To do work in AI, it is important to have advanced math skills.	106	3.3	.71	.16
AI is accessible to anyone who wishes to learn about it.	106	2.9	.86	.57
To do work in AI, it is important to have a brilliant mind.	106	2.6	.92	.93
AI is cool.	106	3.5	.69	.40
AI is a male-dominated discipline.	106	3.4	.81	.29
Work in AI will have a large societal impact.	105	3.9	.35	.99

4-point scales from 1-Strongly Disagree to 4-Strongly Agree

Many of the preconceptions of cybersecurity noted by Ojha et al. were affirmed by participants in our survey (Table 1). Some negative preconceptions (difficult/time consuming/requires advanced math skills/male dominated) were widely agreed with, with a few more positive preconceptions (AI is cool/will have a large societal impact) also affirmed. That AI is "accessible to anybody who wishes to learn about it" found tepid agreement, with a mean slightly above the midpoint of 2.5, indicating that there is work to be done in making AI seem accessible to new students. Responses to the negative misconception that AI requires a "brilliant mind" were mixed. No preconception was significantly more prevalent among women or men. However, this doesn't mean that widely reported preconceptions impact men and women's motivation to study AI in the same way.

One potential method of combating negative preconceptions of AI prior to self-selection into sub-specializations, is by incorporating AI education into non-AI computing courses. A model of this is already present in responsible computing, which has been integrated into introductory programming courses with some degree of success [4, 5]. The can be an increasingly salient option as universities are struggling to hire faculty and offer coursework in AI as part of a recent AI "gold rush" described in Inside Higher-Ed's July 2023 report [2]. We recommend that instructors consider a variety of holistic ways to introduce AI into their courses and curriculum.

REFERENCES

[1] A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence - Michael Haenlein, Andreas Kaplan, 2019: Accessed: 2023-10-03.

[2] Colleges Race to Hire and Build Amid AI 'Gold Rush': *https://www.insidehighered.com/news/tech-innovation/artificial-intelligence/2023/05/19/colleges-race-hire-and-build-amid-ai-gold.* Accessed: 2023-07-30.

[3] Computing Classification System: *https://dl.acm.org/ccs.* Accessed: 2023-07-20.

[4] Doore, S.A. et al. 2020. Assignments that Blend Ethics and Technology. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland OR USA, Feb. 2020), 475–476.

[5] Fiesler, C. et al. 2021. Integrating Ethics into Introductory Programming Classes. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (New York, NY, USA, Mar. 2021), 1027–1033.

[6] Hassani, H. et al. 2020. Artificial Intelligence (AI) or Intelligence Augmentation (IA): What Is the Future? *AI.* 1, 2 (Jun. 2020), 143–155. DOI:https://doi.org/10.3390/ai1020008.

[7] Khogali, H.O. and Mekid, S. 2023. The blended future of automation and AI: Examining some long-term societal and ethical impact features. *Technology in Society.* 73, (May 2023), 102232. DOI:https://doi.org/10.1016/j.techsoc.2023.102232.

[8] Leslie, S.-J. et al. 2015. Expectations of brilliance underlie gender distributions across academic disciplines. *Science.* 347, 6219 (Jan. 2015), 262–265. DOI:https://doi.org/10.1126/science.1261375.

[9] Lewis, C.M. et al. 2011. Deciding to major in computer science: a grounded theory of students' self-assessment of ability. *Proceedings of the seventh international workshop on Computing education research* (New York, NY, USA, Aug. 2011), 3–10.

[10] Nadler, J.T. et al. 2015. Stuck in the Middle: The Use and Interpretation of Mid-Points in Items on Questionnaires. *The Journal of General Psychology.* 142, 2 (Apr. 2015), 71–89. DOI:https://doi.org/10.1080/00221309.2014.994590.

[11] Ojha, V. et al. 2023. Computing Specializations: Perceptions of AI and Cybersecurity Among CS Students. (2023).

[12] Selenko, E. et al. 2022. Artificial Intelligence and the Future of Work: A Functional-Identity Perspective. *Current Directions in Psychological Science.* 31, 3 (Jun. 2022), 272–279. DOI:https://doi.org/10.1177/09637214221091823.

[13] Tai, M.C.-T. 2020. The impact of artificial intelligence on human society and bioethics. *Tzu-Chi Medical Journal.* 32, 4 (Aug. 2020), 339–343. DOI:https://doi.org/10.4103/tcmj.tcmj_71_20.

Exploring Technology for Accessibility Through a First-Year Writing Seminar

Pamela Cutter
Computer Science Department
Kalamazoo College
Kalamazoo, MI, USA
Pam.Cutter@kzoo.edu

ABSTRACT

Technology has become an indispensable part of daily life, yet the number of individuals with the knowledge and skills to advocate for and develop technology with accessibility in mind remains small. Introducing accessibility best practices to students early in their college career provides them the opportunity to learn about resources and strategies that they can carry into their jobs or other activities to improve digital accessibility in their communities. This poster describes a first-year seminar at Kalamazoo College that focuses on technology for accessibility as the basis for engaging in discussion, learning about college resources and college life, and for improving students' critical thinking and expository writing skills. The structure and content of this seminar could be easily replicated or modified to work with other models of first-year seminars.

CCS CONCEPTS

• Human-Centered Computing ~ Accessibility • Social and Professional Topics ~ Professional Topics ~ Computing Profession ~ Assistive Technologies

KEYWORDS

Accessibility; First-year seminar

ACM Reference format:

Pamela Cutter. 2024. Exploring Technology for Accessibility Through a First-Year Writing Seminar. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/ 3626253.3635596

1 INTRODUCTION

First-year seminars are an integral part of the curriculum at most colleges and universities throughout the United States. Addressing topics such as an introduction to college resources, time management, learning strategies, and basic study skills, the most common types of seminars also aim to improve students' critical thinking and expository writing skills. In addition to orienting students to college life, the first-year seminars at Kalamazoo College aim to engage students in the study of significant social and historical differences. Faculty from across the college take an idea, event, theme, or question as their organizing principle and work at the edges, where areas of knowledge intersect with each other and with the "real world" of college students. Starting their first term in college, students are challenged to look beyond themselves, to learn about and respect others who may be different from themselves, and to learn to apply their skills to make their communities and their world a more inclusive space.

My goal was to re-design a first-year seminar that I had previously taught with a focus on gender and race in science to now focus on the topic of technology and accessibility. While some institutions ([10]) integrate accessibility throughout their computing curriculum or offer standalone courses on accessibility ([3], [2], [4]), it is becoming more common to include accessibility content in already existing courses ([5], [7], [1], [9]). Re-focusing an already existing first-year seminar on technology and accessibility is another pathway for introducing disability awareness and accessibility skills to help prepare students to create technology that everyone can use. According to the World Report on Disability ([11]), about 15% of the world's population is living with a disability. Nine out of ten American adults use the internet, and nearly one-in-five have a disability that may affect their ability to use the internet. ([6], [8], [11]). Incoming college students have grown up using technology but few of them have thought about how that technology works or who it was designed for, much less who it was not designed for and how they can play a role in advocating for digital accessibility in whatever discipline they study. The seminar introduces them to a new perspective of technology and gives them strategies and resources to carry into their jobs or other activities to improve digital accessibility in their communities.

2 COURSE OVERVIEW

Incoming students at Kalamazoo College select their first-year seminar based on their own interests during the summer before matriculating. Students at Kalamazoo College do not declare a major until halfway through their sophomore year, so the students in any given seminar represent a cross-section of many

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635596

different majors each of whom has expressed an interest in that seminar among the others. Seminars are capped at 16 students to encourage engagement in discussion, interaction with a faculty member, and attention to individual learners' needs. Seminars all meet at the same time, for 75 minutes, 3 times a week, for 10 weeks.

In my first-year seminar exploring technology for accessibility, students learn about accessibility in the context of computer usage and study the distinction between accessibility, usability, and inclusion. Topics explored include the following:

- What constitutes a disability
- Barriers faced by those with disabilities
- Accessibility as a human right
- Assistive and adaptive technologies
- Guidelines and principles for designing and developing accessible systems and technologies (including universal design and Web Content Accessibility Guidelines)
- Americans with Disabilities Act (including financial and legal obligations)
- Becoming a digital ally and advocate for those with disabilities

Small group discussions of 3-4 students give each student the opportunity to explore the different group roles of task manager, notetaker, and reporter, as well as challenge them to expand their previous knowledge of accessibility. Other activities include analyzing websites and mobile apps using web content and universal design guidelines, exploring barriers and accommodations for individuals with disabilities on our campus, meeting with college administrators such as the Chief Information Officer, the Senior Associate Dean of Students, and the Associate Vice President for Human Resources to discuss the ADA and accommodations that our campus provides, and meeting with local agencies to learn about assistive and adaptive technologies that are available for individuals in the community with disabilities. These provide a solid foundation for students to explore through writing. Students in the seminar write weekly structured reflections, along with short response papers on topics we cover in discussions or on activities they have engaged in. They also participate in a library research session on the topic of AI and accessibility which culminates in an annotated bibliography and presentation.

3 CONTRIBUTIONS

This first-year seminar not only acclimates new college students to college life, resources, and expectations, but does so in the context of technology and accessibility. It gives students the opportunity to learn more about disabilities as well as the guidelines and tools for digital accessibility and the role they can play in becoming allies and advocates for disabled individuals in whatever field they choose to pursue.

Course materials will be posted to UC Boulder's Promoting the Integration of Universal Design into University Curriculum (UDUC) database for use by other faculty.

ACKNOWLEDGMENTS

This work was supported by a Teach Access Faculty Grant. (https://teachaccess.org/initiatives/faculty-grants/).

REFERENCES

[1] Jaskaran Singh Bhatia, Parthasarathy P.D., Snigdha Tiwari, Dhruv Nagpal, and Swaroop Joshi. 2023. Integrating Accessibility in a Mobile App Development Course. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education* V. 1 (SIGCSE '23) March 2023. 1021-1027. DOI: https://doi.org/10.1145/3545945.3569825

[2] CSC575: Accessible Computing Seminar. (https://home.adelphi.edu/~siegfried/cs575) [Accessed 10 July, 2023]

[3] Yasmine N. Elglaly. 2020. Teaching Accessibility to Software Engineering Students. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (SIGCSE '20). March 2020. 121-127. DOI: https://doi.org/10.1145/3328778.3366914

[4] ENGR302: Accessible Design. http://catalog.wichita.edu/undergraduate/courses/engr/ [Accessed 30 Sept., 2023]

[5] Mexhid Ferati and Bahtijar Vogel. 2020. Accessibility in Web Development Courses: A Case Study. *Informatics*. 2020 7(1):8. DOI: https://doi.org/10.3390/informatics7010008

[6] Internet/Broadband Fact Sheet, Pew Research Center, 2018. http://www.pewinternet.org/fact-sheet/internet-broadband/ [Accessed 30 March, 2023]

[7] Lin Jia, Yasmine N. Elglaly, Catherine M. Baker, and Kristen Shinohara. 2021. Infusing Accessibility into programming courses. In *Proceedings of the Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. May 2021. Article No. 231. 1-6. DOI: https://doi.org/10.1145/3411763.3451625

[8] Nearly 1 in 5 People Have a Disability in the U.S., Census Bureau Reports, 2012. https://www.census.gov/newsroom/releases/archives/miscellaneous/cb12-134.html [Accessed 30 March, 2023]

[9] Chia-En Tseng, Seoung Ho Jung, Yasmine N. Elglaly, Yudong Liu, and Stephanie Ludi. 2022. Exploration on Integrating Accessibility into an AI Course. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*. V.1. February 2022. 864-870. DOI: https://doi.org/10.1145/3478431.3499399

[10] Annalu Waller, Vicki L. Hanson, and David Sloan. 2009. Including accessibility within and beyond undergraduate computing courses. In *Proceedings of the 11th International ACM SIGACCESS Conference in Computers and Accessibility*. 155-162. DOI: https://doi.org/10.1145/1639642.1639670

[11] World Report on Disability. 2011. World Health Organization. https://www.who.int/disabilities/world_report/2011/report/en [Accessed 30 July 2023]

Centering Ethics in Computing Student's Mind: A Question Directed Approach for Teaching and Learning Tech Ethics

Udayan Das
Mathematics and Computer Science
Saint Mary's College of California
Moraga, CA
udd1@stmarys-ca.edu

ABSTRACT

There continues to be growing interest in including ethics in undergraduate computer science and engineering education. While this impulse is greatly welcome, the subject of ethics and how it relates to the profession and professional practice can be somewhat abstract. The style of teaching and learning about ethics varies from theoretical philosophy courses to courses that carefully examine the harms and impacts of systems and technologies. Students do not necessarily leave with models of how to continue thinking about ethics in the profession or a framework that they can easily apply in the future. This poster presents a question directed framework that is used in teaching of a tech ethics course. The questions are easy to understand and repeat instruction makes it natural for students to always use the questions to guide their thinking towards ethical matters throughout the course. The questions are memorable, and it is expected that students will continue applying this ethical framework in the future as they step into a profession in which they will inevitably have to grapple with ethical questions due to how closely our profession is tied to human life now and in the future. The framework also specifically addresses the how of ethical professional practice which teaches students about how to build community and social capital in the workplace and thus make it easier to advocate for ethical behavior.

KEYWORDS

CS education, ethics education, engineering ethics, tech ethics, impact of computing on society, ethical advocacy, responsible computing, professional ethics

ACM Reference Format:
Udayan Das. 2024. Centering Ethics in Computing Student's Mind: A Question Directed Approach for Teaching and Learning Tech Ethics. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635496

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635496

1 INTRODUCTION

As undergraduate computer science (CS) and engineering education programs continue to evolve, the urgency of ethics in education is being recognized. This can be seen in the various tech ethics course offerings at educational institutions [1] and the inclusion of ethics requirements in undergraduate CS and engineering programs.

The past couple of years has also seen the coming together of CS educators around the issue of thinking about and developing guidelines for the Teaching of Responsible Computing [2]. While these are welcome developments the style of teaching ethics tends to stay at one of two extremes. A course heavily focused on the philosophy of ethics and a course that spends a great deal of time considering the impacts and harms of technology, particularly for specific application areas such as AI [1]. Both approaches are valuable in their own way, but a concern for this instructor has been that neither of these approaches equips students with how to actively engage in ethics throughout their future careers. Towards addressing this gap, this poster presents an approach that uses a question directed approach for carefully examining the various dimensions of tech ethics and to provide students with a memorable framework that they can continue to apply in their future professional life.

2 WHY A QUESTION DIRECTED APPROACH?

When I began teaching tech ethics, I was daunted by the prospect of having to teach aspects of moral philosophy. I was also concerned that students who learn about philosophical models such as Utilitarianism or Golden Rule do not necessarily retain these in the long run, nor explicitly apply models they have learned when they face ethical problems outside the context of an ethics class. Conversely, the other dominant model of teaching tech ethics relied heavily on discussions of harms and impacts, which, while valuable, does not necessarily equip students with a systematic framework for how to think about situations in the future.

After having taught tech ethics once using a standard model that began with a discussion of philosophical models and then spent time of several case studies of impacts and harms, I switched to a new course design that used questions as a means of making thinking about ethics more accessible, while capturing the complexity of considerations that are inevitably involved.

The question directed approach is presented as it is used in the next section. Here, I will briefly discuss each question group:

- **Why?** This is of course top of mind when students are asked to take this course. In both institutions where I have taught the tech ethics course it is a required course and I think it is important to motivate the need for why thinking about ethics matters. Why should anyone care about ethics? Let's walk through that.
- **Who?** It is important that students learn that there are various stakeholders in each situation and that they have different perspectives and concerns. This will not only make them better advocates for ethical conduct, but also enable them to practice empathy and take others views into account when thinking about solutions.
- **What?** This is of course crucial. This part of the framework carefully addresses the spectrums of concerns and the various impacts and harms.
- **Where?** One of the wordless messages conveyed when ethics is *pulled out* and taught in its own course is that ethics may seem like an afterthought. This question area is to get students to think about ethics in every aspect of their professional practice. (Note: the question framework is also used in a shorter version in the intro programming course, and in depth in a capstone course. Thus, the training extends beyond this course alone.)
- **How?** A question that is in my experience universally not addressed is how to negotiate the workplace and advocate for ethical behavior. This is particularly critical for early career professionals who may not have the power in the workplace to be able to speak up for or effectuate ethical behavior. This part also explicitly brings in professional codes such as ACM [3].

3 THE QUESTION DIRECTED TEACHING AND LEARNING MODEL: THE WHY, WHO, WHAT, WHERE, HOW, AND WHEN OF TECH ETHICS

1. **WHY** care and worry about ethics
   a. Why do ethics matter?
   b. Why do you care about ethics? Do you?
   c. Impact of Technology on Society: Personal, Political, Environmental, Economic
   d. Legal and policy dimensions
2. **WHO** are stakeholders in ethics?
   a. Users
   b. Tech professionals / Tech Workers
   c. Companies
   d. Institutions including governments
   e. Threat actors
3. **WHAT** are the spectrums of Ethical concerns?
   a. Privacy and security
   b. Ownership and access
   c. Surveillance and freedom AND/OR Freedom and Anti-Freedom
   d. Abuses and unintended consequences
   e. Fairness and Justice
   f. Identifying concerns by stakeholder group (connects to 2 above)
4. **WHERE** do tech ethics apply?
   a. Ethics in Design, Development, Data, Deployment, and your Day job
   b. Ethics at home, company/institution, government
   c. Ethics is important in every step of the process and in every setting
5. **HOW** can tech ethics be applied?
   a. Repeat: Ethics in Design, Development, Data, Deployment, and your Day job
   b. At the company: speaking up; advocating for ethics as an employee
   c. At the company: building social capital to help you speak up
   d. At the company: building community
   e. Ethics not as an afterthought but centered in all your processes
   f. Assessing your own impact
   g. How to connect personal ethics to professional ethics
   h. Codes of conduct
   i. Approaches in ethical tech: Design Justice, Values Sensitive Design, Privacy by Design, Consentful Tech
   j. How to use your ethical outlook: thinking beyond reducing harm to actively advancing justice and equity while addressing problems you care about

Bonus: WHEN? All the time

Students are expected to apply the questions to thinking about the case studies provided throughout the semester. Ex: when working through a particular case study they must identify who are the different stakeholders involved and what are the concerns for each group and how they may conflict with each other.

4 CONCLUSIONS

This poster presents an approach that uses a question directed approach towards the teaching and learning of tech ethics. The questions are memorable, and the expectation is that students through repeated instruction in the framework will continue to have this front of mind and therefore consider ethics in all aspects of their future life and profession. In building a better profession it is necessary that we train a community that treats ethics not as an afterthought, but as something central to their professional practice.

REFERENCES

[1] C. Fiesler, N. Garrett, and N. Beard, "What Do We Teach When We Teach Tech Ethics?: A Syllabi Analysis," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, Portland OR USA: ACM, Feb. 2020, pp. 289–295. doi: 10.1145/3328778.3366825.

[2] "Teaching Responsible Computing Playbook," Mozilla Foundation. Accessed: Oct. 11, 2023. [Online]. Available: https://foundation.mozilla.org/en/what-we-fund/awards/teaching-responsible-computing-playbook/

[3] "ACM Code of Ethics and Professional Conduct." Accessed: 10/11/23. [Online]. https://www.acm.org/code-of-ethics

Reducing Barriers to Entry by Removing Prerequisites for a CS1 Introductory Programming Course

Udayan Das
Mathematics and Computer Science
Saint Mary's College of California
Moraga, CA
udd1@stmarys-ca.edu

Chris Fulton
School of Continuing and Professional Studies
Loyola University Chicago
Chicago, IL
cfulton@luc.edu

ABSTRACT

Introductory programming has evolved in many places to become a CS0 course, enabling students to get their feet wet with programming without completing significant math coursework. A scan of CS programs shows that a majority of CS1 programming courses that count towards an undergraduate CS degree continue to have a math or CS0 prerequisite. This experience report discusses the impact of removing the math prerequisite at an R2 university and a small liberal arts college. Having minimal prerequisites has beneficial effects in terms of diversifying the CS student body as well as enabling students to begin CS coursework early, often in the first semester, potentially impacting persistence, but also enabling students to decide, early, if CS is right for them. The high success rate of students of various backgrounds taking CS certificates and pursuing graduate school also shows that aggressive prerequisites in the past may have been functioning as barriers to entering CS programs. If we are serious about supporting diversity, we need to acknowledge the wide disparity in high school education nationwide and that prerequisites are perhaps functioning as a needless barrier. Where the CS0 course doesn't count towards a degree, or there isn't space for that requirement in the program, it is also worth considering whether the CS0 prerequisite is necessary.

KEYWORDS

CS1, Intro Programming, Diversity, Barrier reduction, Broadening Participation in Computing (BPC)

ACM Reference Format:
Udayan Das and Chris Fulton. 2024. Reducing Barriers to Entry by Removing Prerequisites for a CS1 Introductory Programming Course. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635492

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635492

1 INTRODUCTION

Prior to Fall 2018, the CS1 introductory programming course at an R2 university had calculus as a prerequisite. The same was true of a similar course at a liberal arts college until Spring 2022. This poster details the experience of having removed the calculus prerequisite and requiring minimal prerequisites for the CS1 programming course. Author 1 was the program director at the R2 university when the change was put in place and has since moved to the liberal arts college as of Fall 2021. Author 2 is the current program director at the R2 who has left the minimal prerequisite requirement for the CS1 course as is based upon their prior experience with similar requirements at a large public R1 university and a technical college offering associate degrees. Several major types of academic institutions are thus covered by our collective experience, and this demonstrates that most students are prepared to study programming at a CS1 level right away provided there are appropriate teaching and learning methodologies applied. At the R2 and the liberal arts college the change to the prerequisite was coupled with peer tutoring; active learning; project-based learning; and strong student support through academic advising, course touchpoints, and student success coaches.

Evolution. There is an evolution in progress as far as what prerequisites are required to begin a CS1 programming course. In a survey scan, we found that 30% of institutions have limited or no prerequisites beyond high school or GED completion required for students starting a CS1. This no prerequisite approach is consistent with our experience and the purpose of this experience report is to share information with those programs that still have prerequisites, especially calculus prerequisites, and to encourage them to reconsider those requirements.

Impact. There has been a significant increase in students with no calculus background opting to take the course and when students are taking the course. Both authors have strongly encouraged students taking the CS1 course as early as possible and the now almost all students take the CS1 course in the first 2 semesters, with the vast majority taking it in the first semester. (At the R2 the semester is split into two 8-week terms.) Although this poster does not focus on graduate students from non-CS backgrounds, it is worth noting that at the R2 those students in the CS and CS-adjacent masters programs often take the CS1 course as prep for beginning graduate study. There, as in a

majority of graduate CS and CS-adjacent programs there is no explicit undergraduate calculus requirement to begin graduate study in CS. This is a significant observation and worthy of reflection by all CS faculty as to the necessity of the prerequisites at the undergraduate level. We believe this also supports our observation that student motivation is the biggest factor in ensuring student success in the CS1 programming course.

CS1 programming topics. We use Tew et al. [1], curriculum 2013, and curriculum 2023 beta [2] as guides for a set of topics common to a CS1 programming course.

Table 1: CS1 topic areas

Variables
Data types
Console I/O
Expressions
Branching and selection
Loops
Functions
File I/O
Modules and libraries
Objects
Recursion

Scan results. We conducted a scan of CS programs based on publicly available information for 50 programs in the US. These programs included many top CS programs as well as academic institutions of various types, including Carnegie Classified R1 and R2 universities, private and state universities, liberal arts colleges, as well as community colleges.

Table 2: Scan Results of CS programs scan

CS1 course Prerequisite	Percentage
None	30%
Calculus or precalculus	24%
Calc. or precalc. Co-req	12%
College algebra or math reasoning	12%
CS0	30%

* Note: numbers do not add up to 100 because listed requirements sometimes overlap.

2 INSTRUCTOR REFLECTIONS

Instructor 1 reflection. Key points:

- Primarily adult learners at R2.
- Non-profit training experience with underserved individuals taught me that those with little or no background could be brought up to speed as far as programming is concerned with the right teaching and learning approaches bolstered by the appropriate student supports.
- Calculus caused hesitation for students starting at R2 and discouraged many from starting the program. Calculus was a daunting prospect for students returning after many years.
- Calculus prerequisite added a course to 3 course certificate.
- Incoming CS graduate students at R2 do not necessarily need calculus and CS department began having those students take course at the continuing studies unit.
- At liberal arts college there was also calculus requirement which was removed. The BA at liberal arts college now does not require calculus and that has encouraged students majoring in English and Anthropology to double major in CS, which would be harder with the stringent math requirements.
- Students from other departments can also pursue 3 course CS certificate or CS minor without the math requirement.
- I think the greatest driver of student success in CS1 is motivation and engagement.
- Bringing in wider preparation also brings wider background of students, enriching overall learning.

Instructor 2 reflection. Key points:

- An advocate for excluding calculus as an entry criterion for CS1 while not eliminating the course.
- Immediate engagement in technical skills motivated students to continue in coursework.
- Course accessibility to non-computer science majors broadened its reach and value for different majors.
- Students from other disciplines found an interest in programming and continued in subsequent courses.
- No prerequisites for CS 1 made it easier for administrators to approve enrollment requests as a general elective if space was available.
- The requirement of prerequisites, especially math courses, deterred students from enrolling in CS1.
- I observed that students at various types of institutions could succeed in CS1 without traditional prerequisites.
- Removing prerequisites for CS1 can attract more returning students without imposing math-related barriers. Adult learners face higher barriers to returning to school compared to traditional students transitioning from high school and a prerequisite can be a barrier for those whose interest is focused on learning the skill of programming.

3 CONCLUSIONS

In our experience, removing all prerequisites but high school completion or GED has had no impact on student performance in the introductory programming CS1 course. The removal of prerequisites has significant benefits in terms of allowing a wider variety of students to take the CS1 course earlier in their programs. The rationale of requiring students to take calculus prior to taking a CS1 course should be reconsidered. Not only is there no dependence of a standard CS1 programming course on calculus, but this requirement can also function as discouragement and a delay in students interested in taking the CS1 programming course. Towards broadening participation in computing and supporting students from a wide variety of backgrounds, bringing differing prior experience a big step would be making it easier for students to begin programming as soon as possible.

REFERENCES

[1] A. E. Tew and M. Guzdial, "Developing a validated assessment of fundamental CS1 concepts," in *Proceedings of the 41st ACM technical symposium on Computer science education*, Milwaukee Wisconsin USA: ACM, Mar. 2010, pp. 97–101. doi: 10.1145/1734263.1734297.

[2] "CS2023 – ACM/IEEE-CS/AAAI Computer Science Curricula." Accessed: Aug. 18, 2023. [Online]. Available: https://csed.acm.org/

An LLM-based Reflection Analysis Tool for Identifying and Addressing Challenging Topics

Nasrin Dehbozorgi
Kennesaw State University
Marietta, GA, USA
dnasrin@kennesaw.edu

Mourya T. Kunuku
Kennesaw State University
Marietta, GA, USA
mkunuku@students.kennesaw.edu

ABSTRACT

Traditional evaluation of students' learning primarily relies on assessing learning outcomes through either summative or formative assessment methods. In these approaches, the primary emphasis is on the students' learning outcome, rather than the learning process. However, assessing the learning process is as important since it allows providing timely feedback which can directly impact students' learning outcomes. One of the known approaches to getting information about the learning process is the use of formative reflection tools, which also help students in developing their meta-cognitive skills. One effective method for formative reflection is the Minute Paper technique which asks students two concise questions after each class session: what they have learned and what challenges they have encountered. While Minute Papers encourage brief responses, the analysis process can become time-consuming as the number of students and class sessions grows. To address this challenge, in this study, we propose a Large Language Model (LLM)-based reflection analysis tool designed to assess the challenging topics students encounter during each class session. This tool suggests additional learning modules for students to study based on the frequency of the challenging topics. To achieve this, the model utilizes a local repository of lecture materials to create query contexts, which are then input into the LLM as prompts. Students are given access to these recommended resources for further learning, and they are encouraged to provide feedback after completing these modules. These data-driven recommended learning resources serve as continuous content delivery channels to foster a deeper understanding of the subjects at hand.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Natural language processing**; **Information extraction**.

KEYWORDS

Automated Reflection Analysis, Natural Language Processing (NLP), Large Language Models (LLM), ChatGPT, CS Education, Learning Outcome

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635601

ACM Reference Format:
Nasrin Dehbozorgi and Mourya T. Kunuku. 2024. An LLM-based Reflection Analysis Tool for Identifying and Addressing Challenging Topics. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635601

1 INTRODUCTION

To prepare graduates for professional practice and ensure student success, knowledge delivery is not sufficient by itself. Research indicates that, in addition to descriptive and procedural knowledge, attention to the learning process is essential to enhance metacognitive skills and deepen one's understanding of the subject matter [4]. Reflection which is a conscious examination of one's cognitive processes provides an opportunity for the learners to contemplate on their learning and deepen their understanding of the subject matter [4],[1], [5]. Educational researchers have emphasized the significance of incorporating reflective practices in higher education during the past decade to enhance students' ability to monitor their own learning and effectively communicate their learning process to educators [4], [2]. Another wing of the student's success is their engagement in the learning activities. According to the National Survey of Student Engagement, substantial attention should be devoted to involving students in educational activities as it has a strong connection to college outcomes [4]. Student engagement is also a key predictor of both learning and personal development [3]. One strategy to foster responsible learning engagement is by incorporating reflective activities to help students monitor their own learning process [4]. Another key factor that enhances students' academic success is the implementation of effective pedagogical practices and the provision of instructional resources. Offering customized content tailored to students' specific needs encourages their active involvement in the learning process. This study introduces a model that interconnects three aforementioned dimensions of student success: reflection, engagement, and instruction. By utilizing the Minute Paper reflection tool, students are prompted to engage in reflective practices at the conclusion of each class session, encouraging them to contemplate what they have learned or found challenging in the class. Advanced LLM and NLP models are employed to analyze students' responses and identify the areas where they face difficulties. Subsequently, based on the course material and students' challenges, a series of learning modules and resources are generated using the ChatGPT API. This tool helps educators to get insights into students' learning trajectories during the course and provide timely support and interventions. Detailed information regarding the methodology is presented in the next section, followed by the conclusion and the plan for future work.

2 RESEARCH METHOD

The proposed model performs the downstream tasks in two main phases: 1)analyzing students' reflection data to extract challenging topics, and 2) generating learning materials based on both course content and identified challenging topics. To analyze students' reflections, the model leverages structured parsers and the Langchain framework, and for the identification and generation of learning materials, it utilizes Retriever-Augmented Generation (RAG) with the ChatGPT API. Langchain is an open-source framework that streamlines the integration of LLMs like GPT into other applications such as information retrieval systems or chatbots. It excels at prompt engineering, by allowing the design and management of inputs (prompts) that guide how language models, such as ChatGPT, process subsequent inputs. Langchain simplifies tasks like information extraction and response generation by facilitating seamless communication with language models. It also provides an interface for integration with language model APIs such as OpenAI's ChatGPT. In the field of NLP, RAG architecture is used to combine the capabilities of generation with retrieval to produce contextually rich and informed responses across a variety of applications such as document summarization and question answering. In the retrieval process, it looks for pertinent documents or text databases to identify relevant documents or text snippets related to a specific query. It meticulously evaluates and acquires pertinent data as a preliminary step before the text generation phase. Subsequently, a generative model (such as GPT-3 or other transformer models) takes the acquired relevant snippets and pertinent data as a foundation to generate responses. Throughout this process, the collected information is "augmented," enabling the model to produce outputs that are not only data-rich but also contextually relevant. RAG empowers the developed model to dynamically integrate pertinent data during generation, which produces more diverse, and contextually relevant outputs. RAG's applicability extends to a wide array of NLP applications that necessitate the incorporation of external knowledge to generate responses such as content creation, chatbot responses, and contextually rich question answers. In this research these technologies were applied to extract topics from student feedback and, in turn, enable the development of tailored educational content designed to address the needs of students (Figure 1).

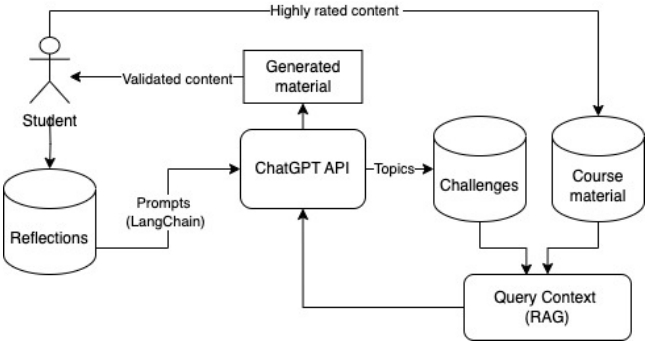


Figure 1: Model Architecture

For this purpose, students' reflections are collected through the "Minute Paper" tool. The raw data is structured via the application of parsers to facilitate further processing tasks. Next, the ChatGPT API is employed to analyze the data and extract challenging subjects. In this step, the Langchain framework is used for prompt engineering and fine-tuning the topics that are extracted from the responses. This process provides more control over the prompts used in the extraction process to produce a curated list of topics that students found challenging. The next step is the generation of additional study material based on the students' feedback. For this purpose, a repository of course material and lecture slides is compiled as the primary knowledge source. In the next step, query contexts are constructed from the knowledge base based on the extracted topics using the RAG framework. This process ensures that the generated material is contextually relevant to the course content as well as students' feedback. To generate resources, the query and its contexts are input into the ChatGPT API. As a result, a set of supplementary materials will be created by the LLM specifically to address students' challenging topics.

3 CONCLUSION AND FUTURE WORK

The objective of this study is to introduce a model that enables instructors to receive real-time feedback on students' comprehension of course content by automating the analysis of reflections. Additionally, the proposed model offers support by generating supplementary learning materials to address students' challenges using LLMs. This generated material is tailored based on the course content and lecture slides. To implement this model, advanced language models, such as LangChain, RAG, and the ChatGPT API, are harnessed to create contextually relevant study resources. The authors intend to assess the model's effectiveness by soliciting student feedback regarding the relevance of suggested study materials to their specific concerns. Through a feedback loop, the model will preserve highly rated resources for future recommendations. One of the limitations of the model that needs further assessment is the validation of generated resources. Currently, instructors are responsible for validation, but future work aims to automate this process. Additionally, future plans involve integrating other LLMs beyond ChatGPT API to access more recent and updated trained data. Overall utilization of this automated reflection analysis tool would have significant benefits for instructors and students in identifying the muddiest points and creating a content delivery channel to address their difficulties.

REFERENCES

[1] David Boud, Rosemary Keogh, and David Walker. 2013. *Reflection: Turning experience into learning.* Routledge.
[2] Nasrin Dehbozorgi and Stephen MacNeil. 2019. Semi-automated analysis of reflections as a continuous course. In *2019 IEEE Frontiers in Education Conference (FIE).* IEEE, 1–5.
[3] Shouping Hu and Alexander C McCormick. 2012. An engagement-based student typology and its relationship to college outcomes. *Research in Higher Education* 53 (2012), 738–754.
[4] Naomi Silver, Matthew Kaplan, Danielle LaVaque-Manty, and Deborah Meizlish. 2023. *Using reflection and metacognition to improve student learning: Across the disciplines, across the academy.* Taylor & Francis.
[5] Aditya Vadapally, Nasrin Dehbozorgi, and Dinesh Chowdary Attota. 2022. Minute-Paper Dashboard: Identification of Learner's Misconceptions Using Topic Modeling on Formative Reflections. In *2022 IEEE Frontiers in Education Conference (FIE).* IEEE, 1–5.

Introducing Theoretical Computer Science Education in Social Sciences and Economics Degrees

Rafael del Vado Vírseda
rdelvado@sip.ucm.es
Universidad Complutense de Madrid
Madrid, Spain

ABSTRACT

The aim of this poster is to present and discuss a curricular methodology to introduce computability topics to social sciences and economics students during the study of their CS subjects in the first courses. These TCS topics give rise to algorithmic impossibility and computational undecidability results which serve to increase students' motivation in their CSE in relation to their university studies. In addition, it allows them to acquire paradoxical and computational thinking skills, with which to deepen and raise computability questions in later courses. To support the impact of the new methodology with evidence, we have analyzed preliminary student results.

CCS CONCEPTS

• **Applied computing** → **Economics**; • **Theory of computation** → **Market equilibria**; **Computability**; *Turing machines.*

KEYWORDS

Computational economics, theoretical computer science, impossibility theorems, undecidability problems, uncomputability

ACM Reference Format:
Rafael del Vado Vírseda. 2024. Introducing Theoretical Computer Science Education in Social Sciences and Economics Degrees. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635481

1 INTRODUCTION AND MOTIVATION

The teaching of *Computer Science* (CS) in the first years of social sciences and economics degrees is often focused on learning the rudiments of programming languages such as *Python*. The learning of concepts related to *Theoretical Computer Science* (TCS) that go beyond the study of the complexity of an algorithm, are usually left out of the CS subjects of these studies. As a consequence, our students finish their *CS Education* (CSE) without having acquired the basic concepts and techniques related to a more theoretical and complete study that properly characterizes CS as a science and discipline. Due to this shortfall, when in later subjects they encounter results such as the *Arrow's impossibility theorem* (which led to the development of social choice and voting theory), or the *undecidability of a computable general equilibrium* (a central issue of the political economy during the Cold War), either their computability implications are not taught, or there are not properly understood. This deficiency in CSE motivates us to believe that the learning of basic TCS concepts and techniques should be introduced and motivated from the first CS courses. To help solve this problem, our aim in this poster is to present TCS activities to social sciences and economics students during their CS subjects in the first courses.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635481

2 RELATED WORK AND CONTRIBUTIONS

CSE in social sciences and economics follows three traditional approaches [4]: The *computational approach* focuses on the use of a particular computer software system or programming language, the *mathematical approach* focuses on data structures and algorithms, and the *applied approach* focuses on how to formulate and solve an economic or social science model with the help of a computer. In this poster, we contribute with a new approach based on *impossibility methods*, that allows us to extend the teaching of CS with TCS concepts through the study of impossibility results. We use impossibility theorems and undecidability results as motivational tools, to introduce TCS content in a gradual way and reason about traditional economics and social sciences problems in novel ways.

3 APPROACH AND METHODOLOGY

As a starting CSE approach, we have recently presented at SIGCSE [1, 2] a successful methodology to introduce TCS content in the early CS courses of science, engineering and technology studies. In this poster, we follow this approach but to develop a new methodology with which to introduce TCS in social sciences and economics studies. To do so, we will start by identifying, within their curricula, *impossibility theorems*, such as the already mentioned Arrow's impossibility theorem, a paradox of social choice showing the impossibility of having an ideal voting structure. Similarly, we start from the *Rabin's impossibility theorem* in economics, on the impossibility to describe the strategic reasoning of competitors in a market. Once students have been intrigued and attracted by these impossibility results, it is the right time to connect, through the progressive design of educational activities in *Python*, the reasons for their impossibilities with classical *problems of undecidability* in TCS. For example, *Gödel's Incompleteness Theorems* for Arrow's impossibility theorem, or the undecidability of *Turing's Halting Problem* for Rabin's impossibility theorem. With these activities, we are able to introduce, in a more natural way, basic TCS concepts and *computational thinking* techniques, such as *Turing machines* in economics and *Rice's theorem* in social sciences.

4 IMPOSSIBILITY THEOREMS

The *Arrow's impossibility theorem* can be presented to our students as a social choice *paradox* that asserts the impossibility of having an ideal voting system, which allows us to address TCS questions related to Gödel's incompleteness and *undecidability as independence from axioms*. In 1950, Kenneth Arrow demonstrated that in any society with at least three possible candidates for each voter, it is *impossible* to provide a social welfare function (or voting procedure) that verifies the following conditions: Non-dictatorship, independence of insignificant alternatives, and unanimity. His assumptions are analogous to *axioms* in modern mathematics, providing a good illustration of the *axiomatic method* and Gödel's *paradoxical thinking*, since it led to the surprising result that no voter can satisfy these conditions. To introduce the Arrow's proof to our CS students, we follow [5] to show by mathematical induction that Arrow's theorem is satisfied if and only if it is verified in the base case, with only two voters and three possibilities, a small amount that allows for a simple demonstration. For this base case, we can verify it by considering the theorem as a *Constraint Satisfaction Problem (CSP)*, and applying a *Depth-First Search (DFS)* algorithm in *Python* to compute all social welfare functions that make the last two conditions true, and finally prove that they all are dictatorial.

5 UNDECIDABILITY PROBLEMS

Around the Arrow's impossibility theorem, we can find motivating *undecidability results*, with important consequences in the economics and social sciences areas [6]. In 1991, the economists Costa, Doria and Tsuji speculate on the application of the Arrow's impossibility theorem to economics, and sketched how to apply the *Halting Problem* of Turing to obtain an undecidability result for competitive markets. Since John von Neumann, the simplest model to understand cooperative behavior has been the *Prisoner's Dilemma*. Our programming practices in *Python* on this game-theoretic model are aimed at learning how undecidability appears even in such affordable models. Following [3], we consider a more realistic version of the classic Prisoner's Dilemma, when it is made open-ended (two players engage in repeated games, never knowing if they will ever meet again). In this iterated version, interesting *competing strategies* emerge and *computational questions* arise among our CS students: When one or the other strategy will evolve to win, or if certain strategies will reach an *equilibrium*. We consider a further dimension, the space: Participants using various strategies compete against their nearest neighbors on a board, represented by cells in different colors. This version of the Prisoner's Dilemma proves undecidable, because strategies progress in a similar way to a *cellular automata* model [3]. We show to our CS students that Turing machines can be embedded in cellular automata, and strategies are able to simulate the behavior of arbitrary Turing machines, thus inheriting the *undecidability of the Halting Problem*.

6 PRELIMINARY RESULTS

We have analyzed the presentation of all these TCS contents from the impossibility results given in **Section 4** and **5**, first, in specific CS subjects through the realization of theory questions, resolution of programming exercises, laboratory practices, and evaluation tests and exams. Second, in specific economic and social science subjects

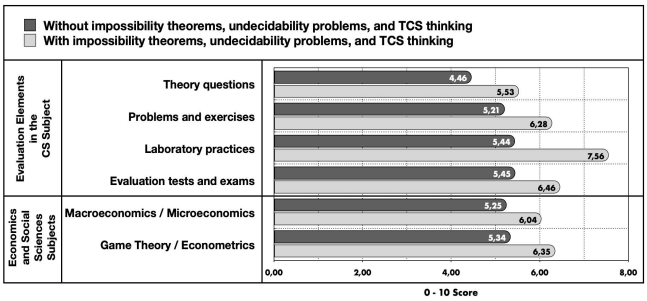


Figure 1: Preliminary results in Social Sciences & Economics.

whose curricula are more related to the TCS content (*Microeconomics*, *Macroeconomics*, *Econometrics*, *Game Theory*), in which our students have been able to address impossibility questions using the TCS skills learned. The educational experience has been executed from 2019-2020 in a Double Degree in Economics & Mathematics and Statistics at a public research university in Spain. Our experience has been applied in two randomly selected groups of students, with the same initial assumptions, and taught by the same teacher. Curriculum topics covered in each group have been common, but in one group, the TCS issues of this poster have been presented during the implantation of the CS subject, while in the other group have not been taught. **Figure 1** shows the *work-in-progress* results achieved in the last course from each group.

7 CONCLUSIONS AND FUTURE WORK

The aim of this poster is to provide CS students of social sciences and economics degrees with TCS tools, with which to reason computationally about traditional problems of these studies in novel ways. In addition, the experience can be extended to CS students in other studies of interest to the SIGCSE community, since the use in traditional TCS courses (e.g., *Automata*, *Computability*) of some of the results presented in the poster (beyond the usual and abstract examples), could contribute to improve the interest of students learning TCS, and enable their adoption by other CSE practitioners. At the request of our CS students, implications for future work will focus on implementing activities to show the consequences that undecidability has on other recent topics (e.g., undecidability conditions of *smart contracts* on *blockchain*).

REFERENCES

[1] Rafael del Vado Vírseda. 2021. Learning from the Impossible: Introducing Theoretical Computer Science in CS Mathematics Courses. In *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education, Virtual Event, USA, March 13-20, 2021*. ACM, 952–958. https://doi.org/10.1145/3408877.3432475

[2] Rafael del Vado Vírseda. 2023. Theoretical Computer Science Education from Impossibility and Undecidability Problems in Physics. In *SIGCSE'23: The 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada). ACM, 270–276. https://doi.org/10.1145/3545945.3569742

[3] P. Grim. 1997. The undecidability of the spatialized prisoner's dilemma. *Theory and Decision 42, 53-80* 42 (1997), 53–80. https://doi.org/10.1023/A:1004959623042

[4] D. A. Kendrick, P. R. Mercado, and H. M. Amman. 2006. Computational Economics: Help for the Underestimated Undergraduate. *Computational Economics* 27 (2006), 261–271. https://doi.org/10.1007/s10614-006-9027-5

[5] P. Tang and F. Lin. 2009. Computer-aided proofs of Arrow's and other impossibility theorems. *Artificial Intelligence* 173, 11 (2009), 1041–1053. https://doi.org/10.1016/j.artint.2009.02.005

[6] K. V. Velupillai. 2009. Uncomputability & undecidability in economic theory. *Appl. Math. Comput.* 215, 4 (2009), 1404–1416. https://doi.org/10.1016/j.amc.2009.04.051

Exploring Student Misconceptions about Concurrency Using the Domain-Specific Programing Language "Sonic Pi"

Giorgio Delzanno
DIBRIS, Univerità di Genova
Genova, Italy
giorgio.delzanno@dibris.unige.it

Giovanna Guerrini
DIBRIS, Univerità di Genova
Genova, Italy
giovanna.guerrini@dibris.unige.it

Daniele Traversaro
DIBRIS, Univerità di Genova
Genova, Italy
daniele.traversaro@dibris.unige.it

ABSTRACT

As the importance of concurrent and multi-threaded programming continues to grow, many universities have incorporated these concepts into their introductory courses. Sonic Pi, a programming language designed for music creation, provides valuable support for exploring concurrency due to its simplified multi-threading abstractions and its domain-specific nature. This paper investigates the combined use of Sonic Pi and Team-Based Learning to mitigate the difficulties in early exposure to concurrency. Sonic Pi provides great support for "playing" with concurrency, and "hearing" common problems such as data races and lack of synchronization among different threads. Our primary research goal is to explore whether the use of Sonic Pi can support students, especially in the early stages, to understand concurrent programming concepts and help them face misconceptions identified in the concurrency education literature. The approach has been applied in teaching experiments with undergraduate students involving 184 participants.

KEYWORDS

Concurrent Programming, Misconceptions, Music and Coding

ACM Reference Format:
Giorgio Delzanno, Giovanna Guerrini, and Daniele Traversaro. 2024. Exploring Student Misconceptions about Concurrency Using the Domain-Specific Programing Language "Sonic Pi". In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635521

1 MOTIVATION AND CONTRIBUTION

Learning to program is a challenging and demanding process because of its close connection to logical reasoning, problem-solving, and creative thinking. One computational concept that introduces further complexity is concurrency, i.e., the execution of multiple tasks at once by means of multiple execution threads working on a shared memory [3]. Concurrency introduces data races and non determinism: concurrent updates could interfere with each other, and the exact order in which execution threads are scheduled may not be determined a priori.

In traditional computer science curricula, concurrent programming was typically introduced at advanced levels. The relevance of concurrency is growing, for instance, multicore computing is eroding Moore's law, and thus several universities are now introducing concurrency in the first programming courses [2, 5]. However, students often struggle to understand concurrency, which can be particularly challenging when introduced early in the curriculum. Consequently, there is a need for research aimed at supporting students in building and refining their viable mental models. In this setting, domain-specific programming languages can play an essential role in reformulating complex concepts in a more natural way.

In this paper, we developed a pedagogical experiment related to introducing concurrency and multi-threading at different stages of the CS degree program. Our approach is based on the use of Sonic Pi [1], originally designed for live music coding performances. The Sonic Pi programming language provides abstractions to introduce basic computational concepts as well as advanced concepts such as multi-threading. Sonic Pi offers distinct advantages over traditional computing methods to facilitate the early introduction of multi-threading. The inherent concurrency of music creation, e.g., an orchestra, and the simplicity of the Sonic Pi editor provide a natural and immersive learning experience for concurrency. In particular, live coding and auditory output, two key features of Sonic Pi, provide great support for "playing" with concurrency and "hearing" data races and other common mistakes.

Based on previous research on concurrency education [5], our specific research questions are: Does the use of Sonic Pi help the students face common misconceptions and mistakes in concurrent programming? Is this more helpful for early-stage programmers? Can we apply knowledge transfer from Sonic Pi to more traditional programming languages such as C and C++? What is the effect in terms of learning outcomes?

To answer these questions, we have designed an introductory approach to concurrency for undergraduate students based on Sonic Pi and the collaborative teaching methodology of Team Based Learning (TBL) [4]. The proposed approach is the result of extensive research and classroom experimentation over the past three years. In particular, the approach has been applied to two teaching experiments for undergraduate CS students in the academic year 2022/23, more precisely, 130 first-year students of the computer architectures course (CA) and 54 third-year students of the concurrent and distributed algorithms course (CDA). By collecting and analyzing team responses and tasks, we delved into the students' misconceptions about concurrency. To our knowledge, this study represents the first attempt to use Sonic Pi for these learning goals at the university level.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635521

```
use_synth :piano
notes=[:E3,:G3,:B3]
counter=0

define :foo do |x|
  in_thread do
    play notes[x]
  end
end

3.times do
  foo counter
  counter+=1
end
```

Voting Cards

Which statement is true?

The program has no race conditions.

The program is single threaded.

The auditory output is a sequence of three notes.

The auditory output is the E minor chord.

Figure 1: Example of Team App task.

2 DETAILS ABOUT THE APPROACH

In the context of the CA and CDA courses, we introduced students to multi-threading through a series of short practical tasks, each designed to address specific misconceptions about concurrency, allowing students to gradually build their understanding while addressing concepts such as thread local variables, race conditions, correctness and termination, and inter-thread synchronization. Some tasks focused on program comprehension, while others involved code writing. The first experiment was conducted just before students in the CA course were introduced to multi-threading in C. This timing allowed students to gain a basic understanding of concurrency through the Sonic Pi-based approach. The second experiment took place at the beginning of the CDA course, where students were introduced to concurrency concepts more in depth. The activities did not contribute to the student's final summative assessment, they only had formative value.

During the activities, we collected RAT quiz scores and team App tasks to analyze students' understanding of concurrency and any misconceptions that might arise. We also collected data through an anonymous individual post-questionnaire on students' appreciation/perception of the activity. Finally, we collected the results of a post-individual test to investigate knowledge transfer. The test consisted of three exercises formulated either in pseudo-code or in the C programming language. These exercises proposed a similar concurrency scenario, but in other languages, and provided practical opportunities for students to apply their understanding of concurrency in a different learning context.

Familiarisation with the TBL methodology and the Sonic Pi language is achieved by means of tutorials assigned to the students one week before the in-class activity. About 2-3 hours of individual study are estimated for the preparation before the class sessions. The RAT quiz consists of five multiple-choice questions that assessed the participants' understanding of the Sonic Pi language syntax and concurrency concepts. Following the TBL methodology, the quiz was completed first individually and then in teams. The team App tasks included a variety of question formats, including multiple-choice questions (voting cards) and open-ended tasks that required students to implement Sonic Pi code. Each exercise focused on specific misconceptions related to concurrent programming. We focused on a range of concurrent concepts, such as data races and synchronization mechanisms, as well as fundamental programming concepts like variable scope and function calls.

An example of task that shows how to use Sonic Pi to "hear" misconceptions is shown in Fig. 1. The global array variable `notes` contains three notes represented by their code (the E note in the 3rd octave, and so on). Function `foo`, with parameter `x`, spawns a thread that plays the `x`th note in the array. The function is called three times in a loop. The `note` array is shared among the threads, however each thread has access to a different cell. When executed, the three notes are played in some casual order by the corresponding threads. In Sonic Pi, the execution of a sequence of `play` instructions is perceived as a single chord. Therefore, the auditory output of the program corresponds to the E minor chord.

3 DATA ANALYSIS AND CONCLUSIONS

To evaluate the effectiveness in terms of learning outcomes, we considered TBL data, in particular team responses to the TBL tasks. Additionally, we gather preliminary empirical evidence on the transferability of the concurrent programming knowledge, skills, and abilities that students acquire in our Sonic Pi-based learning environment to other traditional programming languages, such as C. Knowledge transfer can pose challenges such as differences in the underlying notional machines of the considered programming languages.

To investigate knowledge transfer, we designed a written test consisting of three exercises in either C or pseudo-code, focusing on concurrency aspects covered in the music code-based domain. The test was proposed one week after the experiment. Furthermore, to assess the effectiveness of the activity in terms of engagement and appreciation we considered students' overall perceptions through a post-questionnaire. The misconceptions emerged from the TBL collected data correspond to those identified in the literature, supporting the adequacy of Sonic Pi in this application.

The proposed activities were recognized useful by the students: the median value concerning the evaluation of the usefulness of the proposed activities was 4 on the Likert scale from 1 to 5. In particular, participants found the activity of great help in detecting and learning how to avoid the misconceptions presented in the exercises. Around 80% of the students reported a fair level of difficulty. Exercises using C syntax required more effort to adapt a mental model from one programming language to another with a different notional machine. Further research is however needed to explore knowledge transfer from Sonic Pi to other concurrent languages.

REFERENCES

[1] Samuel Aaron, Alan F. Blackwell, and Pamela Burnard. 2016. The development of Sonic Pi and its use in educational partnerships: Co-creating pedagogies for learning computer programming. *Journal of Music Technology and Education* 9, 1 (3 2016), 75–94. https://doi.org/10.17863/CAM.8369

[2] Daniel Ernst and Daniel Stevenson. 2008. Concurrent CS: preparing students for a multicore world. *ACM SIGCSE Bulletin* 40 (06 2008), 230–234. https://doi.org/10.1145/1384271.1384333

[3] Maurice Herlihy. 1993. A Methodology for Implementing Highly Concurrent Data Objects. *ACM Trans. Program. Lang. Syst.* 15, 5 (November 1993), 745–770. https://doi.org/10.1145/161468.161469

[4] Larry K. Michaelsen, Warren E. Watson, John P. Cragin, and L. Dee Fink. 1982. Team-based learning: A potential solution to the problems of large classes. *Exchange: The Organizational Behavior Teaching Journal* 7, 4 (1982), 18–33.

[5] Filip Strömbäck. 2023. *Teaching and Learning Concurrent Programming in the Shared Memory Model.* Ph. D. Dissertation. Linköping University, Sweden. https://doi.org/10.3384/9789180750011

Style Anomalies Can Suggest Cheating in CS1 Programs

Benjamin Denzler
Computer Science & Engineering
University of California, Riverside
Riverside, CA, USA
bdenz001@ucr.edu

Frank Vahid
Computer Science & Engineering
University of California, Riverside
Riverside, CA, USA
vahid@cs.ucr.edu

Ashley Pang
Computer Science & Engineering
University of California, Riverside
Riverside, CA, USA
apang024@ucr.edu

ABSTRACT

Student cheating on at-home programming assignments is a well-known problem. A key contributor is externally obtained solutions from websites, contractors, and recently generative AI. In our experience, such externally obtained solutions often use coding styles that depart from a class's style, which we call "style anomalies". Examples of style anomalies include using untaught or advanced constructs like pointers or ternary operators or having different indenting or brace usage from the class style. We developed a tool to automatically count style anomalies in student code submissions. We used this tool to find suspected cheating in student submissions for lab assignments across five terms of CS1. This poster presents our findings: Some student submissions were suspected of cheating due to high style anomaly counts and were *not* flagged as suspicious by a code similarity checker. With the rise of externally obtained solutions from websites, contractors, and generative AI, style anomalies may become an important complement to similarity checking for detecting cheating.

CCS CONCEPTS

• Social and professional topics - Professional topics - Computing education - Computing education programs - Computer science education - CS1

KEYWORDS

CS1, plagiarism, cheating, Program Autograders, Program Style

ACM Reference Format:
Benjamin Denzler, Frank Vahid, and Ashley Pang. 2024. Style Anomalies Can Suggest Cheating in CS1 Programs. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635519

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635519

1 BACKGROUND & RELATED WORK

Similarity checking is the most common approach to detect cheating in computer science (CS) programming classes [1, 2, 3]. But students increasingly cheat with external resources that can yield code dissimilar to classmates' code. For example, finding solutions on websites like GitHub [4, 5] or Chegg [6], or letting AI tools like ChatGPT [7] generate code from English specifications. Fortunately, we have noticed that code from external resources often uses style that departs from the style taught in our classes and textbook, which we call *style anomalies*. Figure 1 presents examples of style anomalies. We aim to use style anomaly detection as a complement to similarity detection to help instructors find suspected cheating cases.

Style checking is a common practice: Google publishes a C++ style guide [8] and maintains the open-source style checker cpplint [9], and the standard Python release includes a style guide [10]. Style checking has also been used for cheating detection before: [11] uses style features to determine the likelihood that programs are from the same authors, and [12, 13, 14] also use style features to strengthen similarity checking. In our approach, instructors configure style features that represent the class's typical coding style so that departures from the class style are brought to the instructor's attention, regardless of code similarity and whether the style is "good".

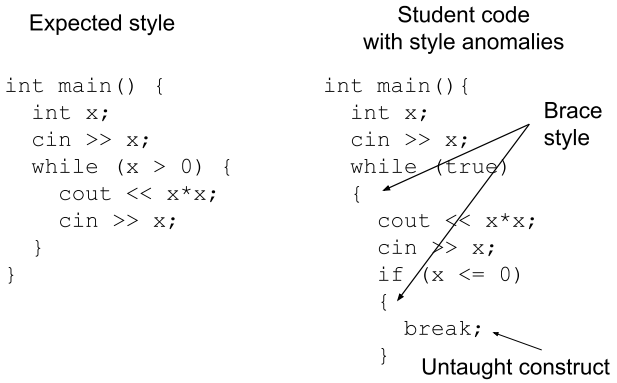


Figure 1: Code from external resources may have style anomalies relative to a class's expected code style.

2 RESEARCH CONTRIBUTIONS

This poster contributes our results of using style anomaly counts to find suspected cheating in student submissions to lab

assignments in CS1. For 6 labs across four terms of CS1 in 2021-2022, we found similarity scores and style anomaly counts for 50 randomly sampled students per lab. We defined a "high" style anomaly count as at or above the 85th percentile of style anomaly counts for an assignment. We defined high counts using percentiles because most students have low style anomaly counts, so students at or above the 85th percentile stand out in the class.

We found that on average, 18% of submissions had high style anomaly counts. More importantly, 8% of submissions on average had high style anomaly counts and were *not* flagged by a similarity checker, meaning these submissions were suspected of cheating but would not be detected by similarity checking alone. We repeated this analysis for a class in Spring 2023 when generative AI like ChatGPT was gaining popularity and found that these numbers rose to 26% and 18%, respectively.

This poster also contributes our findings in using style anomalies to detect code solutions generated by ChatGPT. We asked ChatGPT to generate 18 solutions for CS1 lab assignments. We submitted these solutions to real CS1 labs as a student and found similarity and style anomaly scores for each solution. We found that 44% of ChatGPT's solutions were "high style anomaly" while *none* of the solutions had a high similarity to other students.

These results suggest that as generating code solutions with AI becomes more popular, style anomaly checking will become increasingly important to detect cheating.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant Nos. 2111323 and 2313793.

REFERENCES

[1] Schleimer, S., Wilkerson, D.S. and Aiken, A., 2003, June. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data* (pp. 76-85).
[2] Software plagiarism detector, https://jplag.ipd.kit.edu/, accessed 2022.
[3] Prechelt, L., Malpohl, G. and Philippsen, M. Finding plagiarisms among a set of programs with JPlag. *Journal UCS*, 8(11), 2002.
[4] GitHub.com, accessed August 2022.
[5] Sad CS Major. How UCLA Admins Could Stop The GitHub Cheating and Let Us Get Back To Learning CS, https://medium.com/@joe_bruined/how-ucla-admins-could-stop-the-github-cheating-and-let-us-get-back-to-learning-cs-8f95e8bf6850
[6] Chegg.com, accessed August 2022.
[7] Deng, J. and Lin, Y., 2022. The Benefits and Challenges of ChatGPT: An Overview. *Frontiers in Computing and Intelligent Systems*, 2(2), pp.81-83.
[8] Google's C++ style guide, https://google.github.io/styleguide/cppguide.html, accessed 2022.
[9] cpplint style checker for C++, https://github.com/cpplint/cpplint, accessed 2022.
[10] PEP8 style guide for Python, https://peps.python.org/pep-0008/, accessed 2022.
[11] Oman, P.W. and Cook, C.R., 1989, February. Programming style authorship analysis. In *Proceedings of the 17th conference on ACM Annual Computer Science Conference* (pp. 320-326).
[12] Arabyarmohamady, S., Moradi, H. and Asadpour, M., 2012, November. A coding style-based plagiarism detection. In *Proceedings of 2012 International Conference on Interactive Mobile and Computer Aided Learning (IMCL)* (pp. 180-186). IEEE.
[13] Mirza, O.M., Joy, M. and Cosma, G., 2017, July. Style analysis for source code plagiarism detection—an analysis of a dataset of student coursework. In *2017 IEEE 17th international conference on advanced learning technologies (ICALT)* (pp. 296-297). IEEE.
[14] Karnalim, O., & Kurniawati, G. (2020). Programming style on source code plagiarism and collusion detection. *International Journal of Computing*, 27–38. https://doi.org/10.47839/ijc.19.1.1690

Creation of a CS1 Course with Modern C++ Principles

Ryan E. Dougherty
ryan.dougherty@westpoint.edu
United States Military Academy
West Point, New York, USA

ABSTRACT

Best practices in programming need to be emphasized in a CS1 course as bad student habits persist if not reinforced well. The C++ programming language, although a relatively old language, has been regularly updated with new versions since 2011, on the pace of once every three years. Each new version contains important features that make the C++ language more complex for backwards compatibility, but often introduce new features to make common use cases simpler to implement. This poster contains experiences in designing a CS1 course that uses the C++ programming language that incorporates "modern" versions of the language from the start, as well as recent conferences about the language. Our goals were to prevent many common bad habits among C++ programmers.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; **CS1**.

KEYWORDS

c++ programming language, CS1, introductory programming, modern c++

ACM Reference Format:
Ryan E. Dougherty. 2024. Creation of a CS1 Course with Modern C++ Principles. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635480

1 INTRODUCTION

According to ConsumerReports[1], between 60 and 70 percent of software vulnerabilities are due to memory unsafety. It was also gave examples of languages that perpetuate such unsafety, which include C++. Indeed, the C++ language (among others) is forgiving in letting the programmer do nearly anything, including arbitrary memory accesses. One of the trade-offs is that programs written in C++ compared to other languages, such as Python, are significantly faster. Modern versions of the C++ language within the last decade have introduced new features and enhancements that, although do not remove the previous "unsafe" features for backwards compatibility, allow for programmers to write less such code.

Since the C++ language will be in use for many years to come, any educators who need to teach this language need to do so effectively, especially within a CS1 setting and/or a course taken by non-majors. By "effectively", we mean in a way that promotes not only safe (in terms of memory) programming practices, but in a style that C++ developers have largely adopted and would understand. There are several goals that we should consider. First, students need to be made aware of not only the language itself (and programs within it), but also of these "unsafe" features. Many online tutorials will still contain C++ code with these features, and students may be tempted to follow such examples. Second, educators need to teach a way of programmatic thinking that easily generalizes to larger and more complex problems. C++ is a typed language, unlike Python, and thus adapting existing (typed) code in C++ to work in another scenario involves advanced features, such as templates and inheritance. Although a full understanding of how these features work may be inaccessible to CS1 students, they should have some knowledge of how to *use* these features appropriately. In the context of our institution, the vast majority of students taking the C++ course are non-majors; thus it is vital that we prepare them appropriately as they will likely not be taking another programming course. In this poster we give our experiences in designing a CS1 course involving "modern" C++.

[1]https://advocacy.consumerreports.org/research/report-future-of-memory-safety/

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635480

2 COURSE DECISIONS

We outline the decisions we made in designing a course at Arizona State University in Fall 2018 with modern C++ versions in mind, with a focus on removing any common student bad habits, especially those relating to memory unsafety. There are limited instances where "modern" features can make an entrance in a CS1 course, but our goal was to introduce the mindset of preferring using these features over historic ones when solving "big" problems.

A common source of frustration for CS1 students is with off-by-one errors, and this source sometimes comes from iterating over some linear data structure by index. Consider the simple example in Listing 1, where a raw integer array `a` is created, and then iterates over each of the elements in `a`. In C++11, the range-based `for` loop was introduced, as demonstrated in Listing 2. In our course, we sought to not teach indices for as long as possible.

Listing 1: Simple Iteration Example

```
int a[100];
for (int idx=0; idx < 100; idx++) {
    do_something(a[idx]);
}
```

Listing 2: Ranged-Based For Loop Example

```
int a[100];
for (int elem : a) {
    do_something(elem);
```

```
}
```

At this point in the course, we would introduce students to references, which allow modifying data within a collection in-place. We give two simple choices: one that involves a constant reference to the original object, and another that is modifiable. Both of these choices are demonstrated in Listing 3.

Listing 3: Range-Based For Loop with References

```
int a[100];
// non-modifiable version
for (const int& elem : a) {
    // elem += 1; // does not compile now
    do_something(elem);
}
// modifiable version
for (int& elem : a) {
    elem += 1; // modifies element in place
    do_something(elem);
}
```

C++ is a language that allows for dynamic memory allocation on the heap. Observe Listing 4, which contains an example memory leak. If the `if` statement is taken, then a memory leak occurs. We then advocate using a managed container, such as `std::vector`.

Listing 4: Memory Leak

```
int func() {
    int *a = new int[n];
    ... // modify a
    if (a[0] > 100) {
        return 10; // missing "delete" here.
    }
    delete[] a;
}
```

Unless warnings are turned on, there is another bug that using a managed container solves. Suppose the student accidentally wrote `delete` instead of `delete[]`. We would show students the (optimized) assembly of both versions, proving that the compiler can optimize away the `new`/`delete[]` version of this code, but not the `new`/`delete` version. This has three main benefits for students: (1) there is in fact undefined behavior, (2) there exist different compiler options to help them debug code, and (3) "warnings *are* errors" can be enforced via the `-Werror` flag.

Sometimes using a "raw" pointer is necessary; for example, legacy code or library functions may require them. In C++11, smart pointers were introduced to automatically take care of this problem; the two types are `std::unique_ptr` for unique ownership of an object, and `std::shared_ptr` for shared ownership. The creation of a `std::unique_ptr` object with `new int` passed to its constructor is not inherently wrong in isolation, *assuming the* `std::unique_ptr` *will be fully constructed*. However, consider a slightly generalized example in Listing 5; it contains a function which takes two `std::unique_ptr` objects. If the programmer is using a version of C++ before C++17, there is an unspecified order to the evaluation to the call to `some_func`. Specifically, if the evaluation order is to first dynamically allocate `X`, then call that of `Y`, which throws an exception (i.e., before the `std::unique_ptr` surrounding `X` is constructed), then `X`'s memory will be leaked as there is no mechanism to `delete` it. Since C++17, an argument evaluation order has been standardized.

Listing 5: Unique Pointer Function Example

```
void some_func(std::unique_ptr<X> T1, std::
    unique_ptr<Y> T2) { ... }
...
int main() {
    ...
    some_func(std::unique_ptr<X>(new X), std
        ::unique_ptr<Y>(new Y));
}
```

In the context of our course design, there are several competing objectives. On one hand, this is often the first and only programming course students will take. It would be unwise to give all of the C++ specification details on the behavior of smart pointers and dynamic memory. On the other hand, showing and demonstrating code examples that use `new` may give students false impressions about how modern C++ code is written. Additionally, managing this memory (other than in a small example solely to demonstrate memory leaks) will detract from students' focus on the programming task at-hand.

A common source of student programming bugs is with silent type conversions, especially with converting from a "smaller" numerical type to a "larger" one; we introduce the `auto` keyword to showcase how these bugs can be fixed. This happens fairly early along in the course with the previously discussed motivations, as well as loop examples with `const` reference to avoid copies.

Templates allow for a function, class, etc. to be written effectively once based on one or more (internally held) types, and any variation that use a different internally held type just needs to invoke the template to automatically generate the necessary code. In the context of a CS1 course, templates are not as much of a concept students *implement* directly, but rather *understand* why C++ code operates the way it does. Templates help solve memory unsafety as types (generally) are enforced throughout the function/class.

Functions should not return by `const` value, or by reference (most of the time); the former concerns efficiency, and latter concerns memory unsafety (dangling reference). Of course, other standard coding practices are taught, such as using indentation and curly brackets for ease of reading. Implied previously, we heavily stress using and understanding compiler flags. Specifically to C++, we advocated not using the line `using namespace std;`.

ACKNOWLEDGMENTS

The opinions in the work are solely of the author, and do not necessarily reflect those of the U.S. Army, U.S. Army Research Labs, the U.S. Military Academy, or the Department of Defense.

Designing Theory of Computing Backwards

Ryan E. Dougherty
ryan.dougherty@westpoint.edu
United States Military Academy
West Point, NY, USA

ABSTRACT

The design of any technical Computer Science course must involve its context within the institution's CS program, but also incorporate any new material that is relevant and appropriately accessible to students. In many institutions, theory of computing (ToC) courses within undergraduate CS programs are often placed near the end of the program, and have a very common structure of building off previous sections of the course. The central question behind any such course is "What are the limits of computers?" for various types of computational models. However, what is often intuitive for students about what a "computer" is–a Turing machine–is taught at the end of the course, which necessitates motivation for earlier models. This poster contains our experiences in designing a ToC course that teaches the material effectively "backwards," with pedagogic motivation of instead asking the question "What suitable restrictions can we place on computers to make their problems tractable?" We also give recommendations for future course design.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; • **Theory of computation**;

KEYWORDS

theory of computing, CS course design, CS pedagogy, technical CS course

ACM Reference Format:
Ryan E. Dougherty. 2024. Designing Theory of Computing Backwards. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635479

1 INTRODUCTION & MOTIVATION

The over-arching question that is continually asked in nearly every Theoretical Computer Science (TCS) course is: "what are the limitations of computers?" The motivation is to start with "simple" machines, explore them, and to determine that they are not sufficiently powerful to handle problems that everyday computers can. Then the course would add "computational power" to these machines so that they can more accurately compute answers to problems students should already know how to do with their own machines. Students then learn about decidability and undecidability, and observe that as computer models get more powerful, the fewer questions about them remain decidable, which is a trade-off. One of the main issues with this approach is the justification of *which* "simple" machines to consider initially.

Therefore, we have developed such a course where we have re-thought the commonly taught ordering of topics. In particular, instead of asking the over-arching question located above, we asked a different question: "what *suitable restrictions* on computers can we place?" Here, we start with the computational model that is equivalent to "standard" computers, and immediately find undecidability. Then we explore different restrictions to this model that (re)discover the less-powerful models. In the end, the conclusions and destination are the same–an understanding of computational model limitations–but the exploration is along a completely different road. Here we give our experience in designing a course in this way.

2 COURSE CONTEXT

The course normally involves four main topic sections, taught in the following order: (1) regular languages, (2) context-free languages, (3) Turing machines and decidability, and (4) undecidability. Each section involves constructing a machine or grammar for certain languages of each language type as well as proving that some languages do not fit into the appropriate type. For the Summer 2018 semester, we taught the course differently by instead approaching it from a different perspective. Instead of building up more complex machines from simple ones, we started with complex machines that model everyday computers, and determined what appropriate restrictions one can place on the model that would make certain problems tractable. Specifically, we proposed a new order of the same four topics: (3) → (4) → (1) → (2)[1]. See Figure 1 for a visualization; the black arrows correspond to the standard ordering, and the blue arrows are our proposed ordering. The arrows from nowhere indicates the topic where the associated course starts.

3 COURSE CONSTRUCTION

This section contains our decisions on how we designed our ToC course, which was taught at Arizona State University during the Summer 2018 semester.

Before this course we used the very popular ToC textbook by Sipser [1] as it is very standard, and also contains the concepts in the standard order. However, at the time this course was offered, there did not exist textbooks that taught the concepts in our desired order. Therefore, we needed to design our own lecture notes. We made an important decision in designing these notes in that many of the proofs and examples are intentionally left blank with an empty boxed area for students to fill in the proof/diagram. Each

[1](4) here requires (3) for definitions.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635479

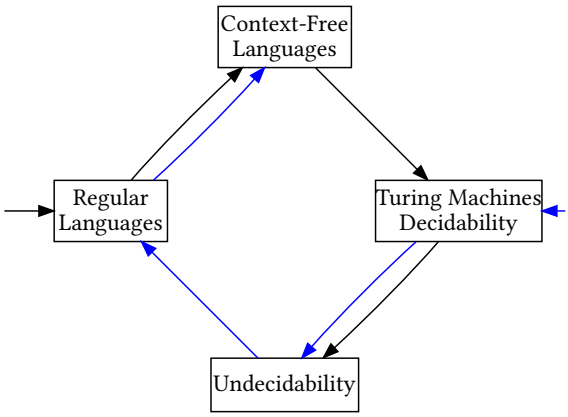


Figure 1: Standard ToC course topic ordering (black arrows) and our proposed topic ordering (blue arrows).

of these blank sections is designed to flow naturally with the rest of the notes as if they were filled in. Some theorems or examples that take less time to lecture upon are built-in to the notes and help provide students how to answer the theorem/example in a way we expect of them.

In the Sipser textbook, the formal definition for Turing Machines involves 7 parts: a set of states, two alphabets (input and tape), a transition function, a start state, and two halting states (for accepting and rejecting). Even the finite automata definition with 5 parts historically was difficult for students to understand. Instead of introducing the formal definition straight away (after motivation), we first do an example on how to build a TM. This choice seems counter-intuitive, in that in mathematical courses the formal definition often comes first with examples to follow. However, we have found that students can more easily derive formal definitions *from* pictures than vice versa. This section otherwise is straightforward, through the comparisons with Turing Machine variants (nondeterministic, multi-tape, etc.), to the Church-Turing thesis.

In the Sipser textbook, decidability and undecidability problems are phrased in terms of questions one can ask about a machine in a given model type. Nearly every non-trivial question one can ask about Turing Machines is undecidable, partly due to Rice's theorem. However, Sipser's textbook contains all such questions grouped together in these sections. An advantage in our approach is that these questions are spread out across the course sections. In lessons we would create a table that contains check marks (for decidable) or "X" marks (for undecidable) for each model and associated question; for Turing Machines, all entries are X marks. As further sections are encountered, more cells are filled in the table. The table would be revisited so that students can make connections with past table entries. We believe it is important to have a compact visual diagram demonstrating the motivation for having different "levels" of models.

We then explain the important perspective they should have, namely in asking what kinds of *restrictions* one can take to the model to make the problems tractable. The Turing Machine model has several behaviors: (1) a read/write one-way infinite tape; (2) a tape head that moves left and right; and (3) a finite set of states with a transition function and some accept/reject criteria. The last cannot be generalized to an infinite set of states nor removed entirely. Therefore, some restriction needs to be made with the first two. A single-tape machine whose tape is write-only is not interesting, and a tape head that only moves left is not interesting as no transitions can be taken without the machine crashing.

Therefore, there are four options: restrict the tape to be read-only, restrict the tape head to move right only, restrict the tape head to move left/right but only at the "end" of where the current work on the tape is, or restrict the tape to be a fixed and finite size (that must of course contain the input). We explain to students that the first is difficult to prove but is nevertheless equivalent to a finite automaton. Students then learn about the machine model for the fourth–linear bounded automata (LBAs)–and the fundamental differences between them and TMs. We then dive into the second option: only moving right on the tape.

Since our Turing Machine model only moves right, the notion of a tape is no longer needed; we then launch into standard terminology and proofs about regular languages and finite automata. The difference is that throughout this the section we again introduce decidability and undecidability now in the context of regular languages. We quickly drop the visuals of a "standard" TM, both in the definition and examples, when defining finite automata as the extra computational power TMs have is unneeded.

We then turn to context-free languages, which are equivalent to a TM that only has read/write operations performed at the "end" of where the current tape contents are. This section is also fairly standard, with again some decidability/undecidability results mixed in with common theorems and examples.

4 FUTURE WORK AND RECOMMENDATIONS

Does this topic ordering result in a statistically significant change in student course performance? Our anecdotal experience indicates that there is not much of a difference, if any; ToC is a fundamentally hard course. Just because the material is taught in a different way, all of the same hurdles need to be overcome.

For practitioners who are looking at changes they want to make to their ToC course, we would strongly recommend to first determine its context within the CS curriculum and whether its needs are currently met. Perhaps the topics themselves need to change; although easily well motivated, automata theory and computability are not the only topics within Theoretical Computer Science. It may be time for the CSE community to reexamine ToC courses and whether they fit modern perspectives of theory and its applications.

ACKNOWLEDGMENTS

The opinions in the work are solely of the author, and do not necessarily reflect those of the U.S. Army, U.S. Army Research Labs, the U.S. Military Academy, or the Department of Defense.

REFERENCES

[1] Michael Sipser. 2012. *Introduction to the Theory of Computation.* Cengage Learning.

Solar Weather, Simulation, and AI in Middle School: Developing a Case Study

Esther Douglass
Hillsdale Public Schools
Hillsdale, NJ, USA
edouglass@hillsdaleschools.com

Katherine G. Herbert
School of Computing
Montclair State University
Montclair, NJ, USA
herbertk@montclair.edu

Vaibhav Anu
School of Computing
Montclair State University
Montclair, NJ, USA
anuv@montclair.edu

Thomas J. Marlowe
Department of Computer Science
Seton Hall University
South Orange, NJ, USA
thomas.marlowe@shu.edu

Sumi Hagiwara
Montclair State University
Montclair, NJ, USA
hagiwaras@monclair.edu

ABSTRACT

In today's rapidly changing technological landscape, it is becoming increasingly evident that solar weather, simulations, and artificial intelligence (AI) are topics that can be and should be integrated into middle school education. Solar weather events can create massive problems with climate, communication, and technology, which can be alleviated by prediction. These events also generate massive amounts of data that AI machine learning can use for such prediction. These topics can be introduced through simulations. How can we integrate solar weather data and simulations with AI-driven models to create interactive and educational software? This paper reviews lessons and pedagogical approaches used to teach about solar weather, AI and incorporates the use of simulations to integrate computer science concepts into other middle school subject areas. The research supports year one of a three-year project. The length of the project allows time to develop a progression to introduce CS concepts and the development of a simulation that can be used in a variety of middle school classrooms.

CCS CONCEPTS

K-12 Computer Science Education, Machine Learning, Neural Networks, Simulations, Artificial Intelligence

KEYWORDS

K-12 Computer Science Education, Data Science, Computer Science, Solar Weather, Machine Learning, Neural Networks, Simulations, and Artificial Intelligence

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635555

ACM Reference format:
Esther Douglass, Katherine G. Herbert, Vaibhav Anu, Thomas J. Marlowe, & Sumi Hagiwara. 2024. Solar Weather, Simulation, and AI in Middle School: Developing a Case Study. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635555

1 BACKGROUND, RELATED WORK, AND LESSON DESIGN

Data science can be used in many different content areas to provide students with a foundation for developing computational thinking with an introduction to neural networks, machine learning, and AI [1,3,4,5]. Solar weather includes a variety of solar events that can impact space weather and, consequently, the Earth. These events include solar flares, coronal mass ejections (CMEs), and solar winds. One important application area is solar weather. Solar weather events can create massive problems with climate, communication, and technology, which can be alleviated by prediction. These events also generate massive amounts of data that AI machine learning can use for this prediction.

This provides an excellent educational opportunity. Introduction to solar weather concepts, large data sets, and fundamental concepts related to AI can increase knowledge of this issue and the underlying science while simultaneously increasing an understanding of CS concepts, including AI and machine learning, data structures (binary trees), and algorithms, and simulation. The value of such lessons and activities was evident during interactions with grant participants of the NSF MSU RET DATA3 program and students at the EpiSTEMic Camp at Montclair State University. Lessons implemented at the camp included an introduction to solar weather, data science, machine learning, and neural networks. During the time spent at the EpiSTEMic Camp, it was apparent that solar weather was a topic new to most students. The Next Generation Science Standards (NGSS) do not address solar weather directly; therefore, it is a topic that can be introduced in various content areas, including middle school CS [1]. In order to assist students in developing a basic understanding of neural networks, ML, and AI, as well as an

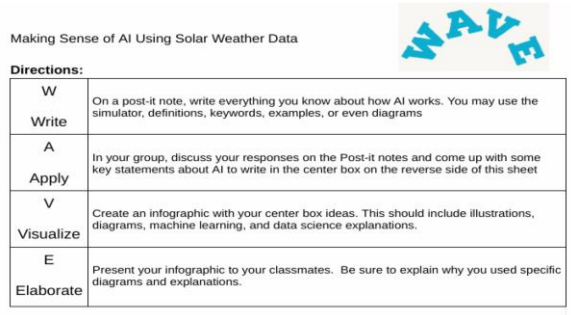
overview of the underlying science and math, lessons, activities, and educational simulations need to be developed [3,4,5].

Solar weather data and simulations offer an opportunity to introduce middle school students to neural networks and AI concepts. Incorporating simulations in the middle school science classroom allows students to interact with virtual environments, conduct experiments, manipulate data, and explore phenomena that cannot be observed directly, such as solar events and exploration of how AI works.

The simulation design has three levels. Level one will be a matching game that introduces students to the characteristics of solar weather using images and data to classify different types of solar weather events. Level two will be a binary tree game that will use the binary tree to train the computer to identify solar weather events and introduce machine learning. Finally, Level three will be the simulator. The simulation will allow the student to input data from solar weather events and observe how AI learns patterns from data, mimics human reasoning, and makes decisions to identify different types of solar weather events. Grant participants of the NSF MSU RET DATA3 program visited NJIT and had opportunities to learn about the massive amounts of data collected through the Earth Cube project [6]. Large data sets used to classify solar weather events can be used as an applicable approach to how neural networks and AI work.

SIMULATOR LESSON: Based on the above experiences, we are guiding students in creating the WAVE experience, which will contribute to the larger simulation project. The larger simulator project will be designed towards the end of the academic year with plans for the project to be designed such that the simulator will help students across multiple academic years contribute and refine each other's work.

The WAVE Process (Figure 1) – Write, Apply, Visualize, and Elaborate – allows students to share their understanding of how ML and AI are necessary to make sense of massive amounts of data collected from solar events. Through various lessons, students will use these steps to make sense of AI using solar weather data. Additionally, these steps conform to NJ State Learning Standards in Computer Science, Career Readiness, Life Literacies & Key Skills, and Science. At the time of submission, students are currently working on the "W" of the WAVE. This lesson will be shared through the Montclair Computer Science Education Hub and Standards Projects, which currently reaches over 50,000 students in northern and central New Jersey schools.


Making Sense of AI Using Solar Weather Data

WAVE

Directions:

W Write	On a post-it note, write everything you know about how AI works. You may use the simulator, definitions, keywords, examples, or even diagrams
A Apply	In your group, discuss your responses on the Post-it notes and come up with some key statements about AI to write in the center box on the reverse side of this sheet
V Visualize	Create an infographic with your center box ideas. This should include illustrations, diagrams, machine learning, and data science explanations.
E Elaborate	Present your infographic to your classmates. Be sure to explain why you used specific diagrams and explanations.

Figure 1: The WAVE Process

FUTURE WORK: Year one lessons are developed to teach about solar weather and an introduction to ML and neural networks. In year two, a simulator will be developed using solar weather data and incorporate the use of AI. During year three, a unit of instruction will be developed and implemented to help students foster a CS skills progression into a better understanding of AI through application.

2 CONCLUSION

There is a need to develop learning opportunities for students to engage in activities that will introduce them to CS concepts, including machine learning, neural networks, data science, and AI. One way to accomplish this is through lessons that can be introduced in various ways and content areas. Solar weather events can interfere with technology and communication; therefore, solar weather provides a pathway for students to explore real-world data and problems that can impact their lives. Understanding the impacts of solar weather events will allow students to make personal connections and make the content more meaningful. In addition, by integrating computing and artificial intelligence, science, and mathematics, students will appreciate the value of applying their knowledge to real-world problems and of an interdisciplinary approach to knowledge.

ACKNOWLEDGMENTS

We acknowledge the support of our fellow MSU RET DATA3 Program members, our colleagues in respective districts and institutions, and our families. This work was funded by the NSF Award #2206885; *NSF RET SITE: Data Sciences and Data Fluency in Scientific Data Sets (DATA3) Program.*

REFERENCES

[1] Camporeale, E. Care, A., and Borovsky, J.E. (2017). Classification of Solar Wind with Machine Learning. *Journal of Geophysical Research: Space Physics*, 122, 910-10,920. https://doi.org/10.1002/2017JAO24383

[2] Martins, R.M., von Wangenheim, C.G., and J.C.R. Hauck (2023)..Machine Learning for All!—Introducing Machine Learning in Middle and High School. International *Journal of Artificial Intelligence in Education*, 2023, https://api.semanticscholar.org/CorpusI DOI:10.1007/s10639-021-10570-8

[3] Vartiainen, H., Tedre, M., and Valtonen, T. (2020). Learning machine learning with very young children: Who is teaching whom? *International Journal of Child-Computer Interaction*, 25, 100182.

[4] http://doi.org/10.1016/j.ijcci.2020.100182

[5] EarthCube data capabilities.bigbear.njit.edu. (n.d.) http://bigbear.njit.edu/ml/

Analogies in Upper Division Computer Science Courses*

Maria R. Ebling
maria.ebling@westpoint.edu
United States Military Academy
West Point, New York, USA

Ryan E. Dougherty
ryan.dougherty@westpoint.edu
United States Military Academy
West Point, New York, USA

Nicholas J. Clark
nicholas.clark@westpoint.edu
United States Military Academy
West Point, New York, USA

ABSTRACT

Analogies have long been proposed as a valuable teaching mechanism, but the question of whether student-generated analogies are more effective for learning compared to those generated by the instructor has not been answered. We compare three different treatments: no analogy, an analogy provided by the instructor, and analogies generated by students. We apply these treatments to two upper-division computer science courses–Operating Systems (OS) and Programming Languages (PL)–and evaluate student learning. Our findings show that any effect of these treatments on student learning is minimal. We found no practical or statistically significant differences between them. Anecdotally, the student-generated analogies provided an active learning activity, and the instructor-provided analogies were the only ones mentioned in written responses to graded events. Given the small differences in learning, instructors can select the treatment based on their needs; for example, selecting the student-generated option for active learning or the instructor-provided option when time is short.

ACM Reference Format:
Maria R. Ebling, Ryan E. Dougherty, and Nicholas J. Clark. 2024. Analogies in Upper Division Computer Science Courses. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635506

1 INTRODUCTION

People tend to learn new concepts by connecting them to those they already know. Many people have studied the use of analogies in teaching computer science, predominantly in introductory courses. Few studies have examined whether instructor-provided analogies or those generated by students result in better learning outcomes, and few studies considered upper-division courses. Curious about the best approach, we designed and executed an experiment to investigate the question of which treatment (student-generated, instructor-provided, or no analogy) yields better learning performance. In this poster, we share the results of a study that evaluates the use of instructor-provided and student-generated analogies, comparing them both to the case where no analogies are used.

*The views expressed here are those of the authors and do not reflect the official policy or position of the Dept. of the Army, Dept. of Defense, or the U.S. Government.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635506

2 BACKGROUND

Analogies have been applied to learning computer science for decades and tend to engender strong opinions [2, 5]. Gentner [4] viewed an analogy as a mapping from a base domain, the one already understood by the student, to the target domain, the one under study. The use of analogies in introductory computer science has been studied extensively [3, 6–9], but only a few studies have examined upper-division computer science courses [1]. Frequently, these studies have not evaluated student learning outcomes and many also do not consider student-generated metaphors. Student-generated metaphors have, however, been studied in fields other than computer science [10]. Our study examines the use of analogies in two upper-level computer science courses typically taken by juniors and seniors at our institution.

3 METHOD

This study examines two courses: Programming Languages (PL) and Operating Systems (OS). Each course had three sections taught by a single instructor. Each instructor picked three lessons whose topics had a natural analogy. We used three treatments: no analogy (N), student-generated analogy (S), and instructor-provided analogy (I). For each lesson, each section was assigned a different treatment. To ensure fairness, we rotated the treatments around the sections across the three relevant lessons. Each section and each student experienced each treatment once; no two sections experienced the same treatment on the same lesson.

In the "no analogy" treatment, we presented information about the topic and then presented a summary without using an analogy. In the "student-generated analogy" treatments, the instructors presented students with a summary of the topic and then asked students to come up with an "everyday" analogy related to the topic; additionally, after giving students several minutes to design their own analogy, they were then asked to share their analogy with a partner. For the "instructor-provided analogy" treatment, the instructors presented information about the topic, and then provided their own analogy. For example, the analogy used for proof by structural induction was that of constructing a building and the analogy for a file system was a library.

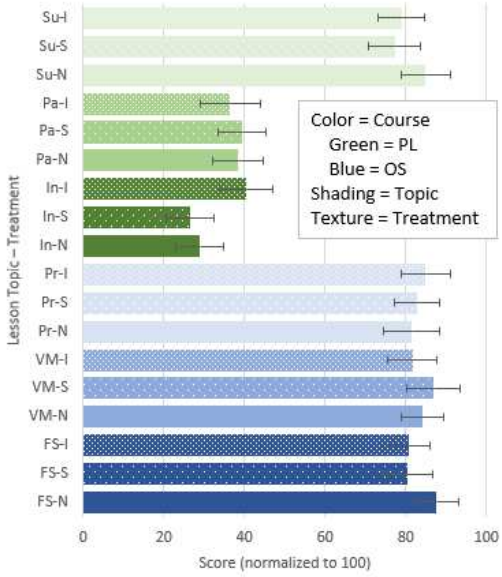
The topics selected for the use of an analogy in Programming Languages were: Recursive Descent Parsing (Pa), Proof by Structural Induction (In), and Subtyping (Su). The topics selected in Operating Systems were: Processes vs. Threads (Pr), Virtual Memory (VM), and File Systems (FS). We measure student learning by looking at student performance on an exam. Because the assessments had different numbers of points, we scale the raw scores.

Table 1 documents the data collected. For each topic, we show the student population in each treatment: no analogy (N), student-generated (S), and instructor-provided (I). We also show the dates of both the lesson and the associated assessment.

Table 1: Dates and Attendance for Lessons and Events

	Topic	N	S	I	Lesson	Exam
PL	Parsing (Pa)	12	12	9	Sept 7	Sept 12
	Induction (In)	13	13	12	Sept 19	Oct 17
	Subtyping (Su)	13	12	13	Dec 2	Dec 13
OS	Processes (Pr)	9	14	12	Sept 7	Sept 26
	Virtual Memory (VM)	17	10	12	Oct 14	Nov 3
	File Systems (FS)	15	12	16	Nov 28	Dec 13

Figure 1: Assessment Scores Across Analogies



4 RESULTS

We performed a mixed-effects multiple regression analysis for the effects of the treatment, the effect of topic, and the interaction between treatment and topic. We examined both a statistical model with main effects and interaction effects and a model containing only the main effects of treatment and topic; the former is preferable (P=0.019). The treatment effect in the full model is not statistically significant (P=0.419) whereas the interaction term (P=0.030) indicates the presence of a cross-over effect suggesting that, between topics, the impact of analogies may be reversed.

The marginal means plots in Figure 1 show the adjusted confidence intervals for each treatment/topic combination. The bar colors represent the course–green for PL and blue for OS–and the different shading indicates the topic. The texture indicates the treatment (heavy dots for instructor-provided, minimal dots for student-generated, and no dots for no analogy).

Statistically significant differences are evident, such as between student-generated and instructor-provided for induction (In). However, the vast majority of confidence intervals within a given topic significantly overlap, suggesting no treatment effect for a given topic. We conclude that the three treatments do not lead to statistically significant differences in learning outcomes. Further, the direction of impact of the effect is not consistent, meaning sometimes having no analogies performed the best and other times instructor-provided analogies performed the best. Table 2 shows the ANOVA.

Table 2: Type 3 ANOVA with Satterwaite's Correction

	Sum Sq	Mean Sq	F value	P(>F)
Treatment	157	78	0.87	0.42
Topic	83190	16638	185.84	< 2e-16 ***
Interaction	1845	184	2.06	0.03 *

We then combine the results from both the student-generated and the instructor-provided analogies to understand the impact *any* analogy had on learning. (Figure omitted due to space.) Once again, sometimes the analogy treatment outperforms the no analogy treatment and sometimes it does not. We observe that none of the results were statistically significant (P=0.41).

Over the course of our experiment, we made two anecdotal observations. First, both instructors noted high levels of student engagement and a diverse set of analogies generated by the students. Second, the instructor-provided analogies were the only ones that appeared in written responses in our assessments.

5 CONCLUSIONS

This study asked whether students learn better when an instructor uses no analogy, provides an analogy, or encourages students to generate analogies. It further examined whether either analogy treatment provided benefit to students. This study examined the use of analogies in two courses taught by two different instructors. It used a sample of convenience, with participants drawn from the ~50 students enrolled in each course. The data show little statistically significant differences between these teaching methods. We conclude that instructors can choose whichever method best aligns with their needs for a given lesson. If the lesson needs an active learning component, the instructor can use student-generated analogies. If instructors have limited time, they can skip the analogy or provide one.

REFERENCES

[1] B. Bettin, L. Ott, and J. Hiebel. 2022. Semaphore or Metaphor? Exploring Concurrent Students' Conceptions of and with Analogy. In *Proc. of 27th ACM Conference on Innovation and Technology in CS Education Vol. 1.* ACM, NY, NY, USA, 200–6.

[2] E.W. Dijkstra. 1988. On the cruelty of really teaching computing science. (Dec 1988). http://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1036.PDF

[3] S. Duvall. 2008. Computer Science Fairy Tales. *J. Comput. Sci. Coll.* 24, 2 (Dec 2008), 98–104.

[4] D. Gentner. 1980. *The Structure of Analogical Models in Science.* Technical Report 4451. BBN, Inc. https://archive.org/details/DTIC_ADA087625/page/n1/mode/2up.

[5] M. Guzdial. 2020. Dijkstra Was Wrong About 'Radical Novelty': Metaphors in CS Education. https://cacm.acm.org/blogs/blog-cacm/248985-dijkstra-was-wrong-about-radical-novelty-metaphors-in-cs-education/fulltext

[6] F. Hermans, A. Swidan, E. Aivaloglou, and M. Smit. 2018. Thinking out of the Box: Comparing Metaphors for Variables in Programming Education. In *Proc. of the 13th Wksp in Primary and Secondary Computing Education.* ACM, NY, NY, USA, 8 pages.

[7] K.N. Macfarlane and B.T. Mynatt. 1988. A Study of an Advance Organizer as a Technique for Teaching Computer Programming Concepts. In *Proc. of the 19th SIGCSE Technical Symposium on CS Education.* ACM, NY, NY, USA, 240–243.

[8] J.P. Sanford, A. Tietz, S. Farooq, S. Guyer, and S.R. Benjamin. 2014. Metaphors We Teach By. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education.* ACM, NY, NY, USA, 585–590.

[9] S. Schez-Sobrino et al. 2019. ANGELA: A Novel Approach of Graphic Notation Based on the Metaphor of Road Signs to Facilitate the Learning of Programming. In *Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality.* ACM, NY, NY, USA, 822–829.

[10] A. Tartaro, B.C. Goess, J. Miller, and J.A. Bui. 2016. Learning Outcomes From a Student-Generated 'Flipped' Wiki Textbook. In *Proceedings of the 19th International Conference on Supporting Group Work.* ACM, NY, NY, USA, 449–452.

The Relationships Between Modality, Peer Instruction Discussion, and Class Sentiment in Hybrid Courses

Salma El Otmani
salma.el.otmani@duke.edu
Duke University
Durham, North Carolina, United States

Janet Jiang
janet.jiang@duke.edu
Duke University
Durham, North Carolina, United States

Shao-Heng Ko
shaoheng.ko@duke.edu
Duke University
Durham, North Carolina, United States

Kristin Stephens-Martinez
ksm@cs.duke.edu
Duke University
Durham, North Carolina, United States

ABSTRACT

Although hybrid courses have become increasingly common in higher education, it remains uncertain whether a student's experience of a course is consistent between in-person and online modalities. To investigate this, we analyzed student modality and discussion data from the Spring 2023 offering of an elective data science course where students are allowed to attend each lecture in person or synchronously online.

Our analysis revealed that there were, counter-intuitively, no statistically significant effects between students' modality mixes alone and their feelings of community and learning in the course. Rather, discussing peer instruction questions with others, particularly while attending class in person, was positively related with stronger feelings of community. However, we did not find a statistically significant relationship between students' discussions and perceptions of whether their modality mix supported their learning.

ACM Reference Format:
Salma El Otmani, Janet Jiang, Shao-Heng Ko, and Kristin Stephens-Martinez. 2024. The Relationships Between Modality, Peer Instruction Discussion, and Class Sentiment in Hybrid Courses. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635514

1 INTRODUCTION

Hybrid courses where students can self-select their modality provide flexibility in how students attend class but present unique challenges [1, 3] due to the simultaneous presence of both in-person and online instruction. It is unclear whether in-person and online modalities are equally as effective in facilitating student engagement in courses where *students can alternate modalities between each lecture* [2]. We thus sought to understand a student's experience in hybrid courses with this level of flexibility.

Our data is from the Spring 2023 offering of a data science elective at a medium-sized, research-oriented private university. The course followed a hybrid format, allowing students to attend each lecture in person or synchronously online.

Our data includes a mid-semester survey and peer instructions (in-class questions) [4]. The mid-semester survey asked students to estimate their modality mix (of in-person and online attendance) and evaluate whether their modality mix supports "[their] feeling of community in the course" and "[their] learning" using a 5-point Likert scale ranging from *strongly agree* to *strongly disagree*. Due to how few students chose some options, we bucketed the options into three categories: *Agree*, *Neutral*, and *Disagree*. All lectures used peer instruction sets (PIs) that were accessible across both in-person and online modalities. The PI data relevant to our research included students' self-estimated modality and whether they engaged in peer discussion between the two rounds of each PI.

Our research question is: *What is the relationship between students' modality and/or peer discussions v.s. their feelings of community and learning?*

2 METHODS

2.1 Data Collection

Students (N=159) filled out the mid-semester survey during weeks 7-8 of the 15-week course. They were asked to self-estimate their modality mix and answer two Likert scale questions.

Table 1: Summary of PI data. Not all PIs had 2 rounds.

Total lectures	# lectures with PIs	# lectures with PIs and modality data	Total PIs	# PIs with 2 rnds	# PIs with 2 rnds and modality data
20	20	19	49	45	42

Table 1 summarizes our PI data. Students were asked to self-estimate their modality for each lecture and, for PIs with a second round, indicate whether they discussed the PI between rounds.

2.2 Metrics

2.2.1 Modality. In studying students' modality mixes, we define a student's *in-person/Classroom*[1] *Modality Fraction* (CMF) as follows:

$$\text{CMF} := \frac{\#\text{ lectures attended } \textit{in classroom}}{\#\text{ lectures attended } \textit{in classroom} \text{ or } \textit{online}}$$

[1]To reduce confusion, in-person is here on referred to as in-classroom.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635514

Question	My ... modality supports my **feeling of community**									My ... modality supports my **learning**								
Response group	Agree			Neutral			Disagree			Agree			Neutral			Disagree		
Metrics	N	Mean	Med.	N	Mean	Med.	N	Mean	Med.	N	Mean	Med.	N	Mean	Med.	N	Mean	Med.
CMF (PI)	96	0.56	0.58	44	0.53	0.56	19	0.43	0.33	124	0.54	0.54	23	0.57	0.73	12	0.49	0.48
CMF (survey)	96	0.60	0.70	44	0.60	0.68	19	0.41	0.23	124	0.58	0.65	23	0.63	0.84	12	0.43	0.27
DF	96	0.75	0.81	44	0.70	0.83	19	0.59	0.64	124	0.71	0.78	23	0.76	0.84	12	0.66	0.64
CDF	93	0.86	0.95	41	0.82	0.93	18	0.71	0.83	120	0.84	0.94	20	0.85	0.91	12	0.72	0.82
ODF	85	0.59	0.65	41	0.48	0.40	19	0.45	0.50	112	0.54	0.59	21	0.54	0.62	12	0.55	0.56

Table 2: Summary statistics of modality/discussion metrics for all categories of student responses to two questions in the mid-semester survey. Not all students have a CDF or ODF depending on their modality mixes.

Metrics	Modality Related	Discussion Related	**Community** K-W Test p	Post-hoc Cliff's δ	Post-hoc p	**Learning** K-W Test p
CMF (PI)	Yes	-	0.21	-	-	0.72
CMF (survey)	Yes	-	0.18	-	-	0.40
DF	-	Yes	**0.033**	0.383	**0.030**	0.48
CDF	Yes	Yes	**0.032**	0.386	**0.027**	0.095
ODF	Yes	Yes	0.15	-	-	1.00

Table 3: Summary of results on relationships between modality-/discussion-related metrics and students' perceptions of how their modality mixes supported their learning/feeling of community. We only report post-hoc results (after Bonferroni corrections) between the Agree and Disagree groups if the Kruskal-Wallis test revealed a significant effect. We found no post-hoc significant effects involving the Neutral group across all tests.

This allows us to compare survey and PI data using a single number for a student's modality mix (as the online complement = 1−CMF).

We assessed the accuracy of students' self-reported modality mix by comparing their CMF from the survey to the PI data up to the survey's due date. Overall, we found that students were fairly accurate in estimating their modality mixes (median absolute difference in CMF = 0.09). However, more students over-estimated the number of lectures they attended in person (67.09%) than over-estimated the number attended online (26.25%).

2.2.2 Discussion. We created the *Discussion Fraction* (DF), *In Classroom Discussion Fraction* (CDF), and *Online Discussion Fraction* (ODF) metrics to measure the frequency of student discussions between rounds of PI with respect to their modality:

$$\text{DF} := \frac{\text{\# PIs discussed}}{\text{\# PIs with 2 rounds}}, \quad \text{CDF} := \frac{\text{\# PIs discussed } \textit{in classroom}}{\text{\# PIs with 2 rounds attended } \textit{in classroom}},$$

$$\text{ODF} := \frac{\text{\# PIs discussed } \textit{online}}{\text{\# PIs with 2 rounds attended } \textit{online}},$$

Note that CDF and ODF are *not* complements and do *not* sum to 1.

3 RESULTS

We investigated how students' behavior (modality and discussing with peers) could have influenced their feeling of community and learning as measured by the Likert questions. As shown in Table 2, students who agreed more with the statement that their modality mix supported their feeling of community generally had higher mean and median values *across all metrics*, implying they *attended more lectures in the classroom, discussed more with their peers, or both.* However, students who responded to be *Neutral* about their modality mix supporting their learning had higher means and medians in most metrics than students who agreed that it did.

To quantify these effects, we conducted Kruskal-Wallis tests for each metric on the three groups (Agree, Neutral, and Disagree) of survey responses. Our null hypothesis was that the metric's underlying distributions are identical across groups. We summarized the results in Table 3.

Counterintuitively, although the two survey questions emphasized *only students' modality mixes*, we found no statistically significant effects for any metric that uses only students' modality. Instead, we found significant effects between students' feelings of community and both the DF (which *only* measured discussion) and CDF (which measured both discussion and modality). For both metrics, post-hoc Dunn's tests revealed a significant, medium-sized effect between the Agree and Disagree groups.

In all, our analyses found substantial evidence that suggests students' *discussions with peers*, particularly while in person, are related to their feeling of community, while revealing that *perceived modality* is intertwined with peer discussion.

4 LIMITATIONS AND FUTURE WORK

A limitation to our results is self-selection bias. Students were allowed to choose their modalities for each lecture, and the sample sizes of each (modality, discussion) combination varied. Additionally, our analyses relied on students' self-estimated modality and discussion data, which may be inaccurate. However, students knew their responses to all questions are ungraded, so we do not expect any inaccuracies to be intentional. Finally, our analysis is based on data from a single semester, as prior semesters lacked comprehensive discussion data. Future research will seek to validate these results with additional data.

ACKNOWLEDGMENTS

We thank the Duke CS+ research program for supporting this work.

REFERENCES

[1] Lucas Kohnke and Benjamin Luke Moorhouse. 2021. Adopting HyFlex in higher education in response to COVID-19: students' perspectives. *Open Learning: The Journal of Open, Distance and e-Learning* 36, 3 (2021), 231–244.

[2] Gerd Kortemeyer, Nora Dittmann-Domenichini, Claudia Schlienger, Ekkehard Spilling, Alina Yaroshchuk, and Günther Dissertori. 2023. Attending lectures in person, hybrid or online—how do students choose, and what about the outcome? *International Journal of Educational Technology in Higher Education* 20, 1 (2023), 1–24.

[3] Natasa Simic, Kristina Mojović Zdravković, and Natalija Ignjatović. 2022. Student engagement in online and face-to-face classes in times of pandemic. 71 (12 2022), 347–362.

[4] Trisha Vickrey, Kaitlyn Rosploch, Reihaneh Rahmanian, Matthew Pilarz, and Marilyne Stains. 2015. Based Implementation of Peer Instruction: A Literature Review. *CBE—Life Sciences Education* 14, 1 (2015), es3.

Hiring, Training, and Managing Undergraduate Teaching Assistants for Large CS1 Classes

Megan Englert*
megan.englert@colorado.edu
University of Colorado Boulder
Boulder, Colorado, USA

Lecia Barker
lecia.barker@colorado.edu
University of Colorado Boulder
Boulder, Colorado, USA

Austin Cory Bart
acbart@udel.edu
University of Delaware
Newark, Delaware, USA

ABSTRACT

As undergraduate computer science enrollments continue to grow, individualized instructor attention becomes increasingly scarce. The impact of social distance between students and their teachers is particularly apparent in large introductory classes, exacerbated by students' lack of common prior experience in computer science. Some institutions remedy class size and experience gaps by hiring advanced undergraduate students as teaching assistants for their introductory courses. However, without the resources to carefully hire, train, and manage undergraduate teaching assistants (uTAs) during the semester, their potential as trustworthy peer mentors and helpful tutors often goes unrealized. This poster presents details of the uTA hiring process, training course, and management strategies used during the 2022-2023 academic year for the introductory computer science courses at the University of Delaware (UD), a large, public, research-focused institution. This system was designed for introductory CS classes, with the goals of low instructor overhead, long-term scalability, and the development of empathetic teaching assistants who could motivate students toward successful learning. Highlights of the system include a hiring process that considers candidates' personality and enthusiasm alongside their technical skills, asynchronous training provided through the university's learning management system, and the use of a head uTA to manage other staff members. Following the implementation of these policies, instructors and uTAs alike reported positive experiences compared to previous semesters without significant changes to students' learning outcomes.

CCS CONCEPTS

• **Social and professional topics → CS1**.

KEYWORDS

CS1, large class sizes, undergraduate teaching assistants, TA training

ACM Reference Format:
Megan Englert, Lecia Barker, and Austin Cory Bart. 2024. Hiring, Training, and Managing Undergraduate Teaching Assistants for Large CS1 Classes. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635613

*Englert was at the University of Delaware at the time of this study.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635613

1 MOTIVATION FOR THE INTERVENTION

Increases in enrollment in undergraduate CS over the past decade have led to large class sizes, which can result in apathy, absenteeism, disengagement, disruptive behavior, and a feeling of disconnection from instructors. These can lower personal accountability within the classroom and increase attrition from the major [3]. Aggravating these problems, the wide variation in prior programming experience can isolate students from their peers [5]. To counteract these threats to classroom culture and student persistence, we modified the introductory course format at the University of Delaware and changed how we hired, trained, and managed undergraduate teaching assistants for this class.

2 COURSE STRUCTURE MODIFICATIONS

Historically, the introduction to CS class was taught in three 50-minute periods per week with lecture and an additional 90-minute lab, with a 60:1 student-to-staff ratio. Much of the course's programming skills were primarily taught through a asynchronous, autograded system, while lectures had students working collaboratively to strengthen their understanding. Labs were staffed by undergraduate TAs. Because labs essentially served as mandatory office hours, attendance declined, as did interaction with uTAs.

To mitigate the variation in prior experience among students even as enrollment grew, the previous course structure was modified. Students are now grouped into 12-person cohorts based on prior experience with CS, major, and level. Within each cohort, no student can be the only person from an underrepresented gender, race, or ethnicity. Each cohort is then matched to a uTA, who leads required practice assignment activity for their weekly assigned lab session. The lab activities are meant to give students additional active learning time while only 12:1 with a member of the course staff.

However, the adapted structure alone could not guarantee that uTAs would be well prepared to lead these active lab activities. Thus, the hiring, training, and management of the CS1 uTAs was overhauled.

3 HIRING, TRAINING, AND MANAGING TAS

Inspiration for hiring and training materials came from a wide variety of other institutions' already existing interview guides, TA handbooks, and peer tutoring resources. Yet, surprisingly little information was available about hiring specifications, interview questions, or important traits to look for in candidates. We therefore created a two-phase hiring process, consisting of a written

application and an interview [1, 4]. Prior to the intervention, uTAs were hired based on their class performance and grades, but knowledge of CS concepts has no bearing on empathy or ability to teach effectively. We therefore added three short-answer questions, now asking about the applicants' most difficult CS course, the qualities they found most important for a uTA, and changes they would like to see for the department and its culture. Each application was scored on a scale from one to five based on thoughtfulness and effort; applications earning high scores demonstrated interest, enthusiasm, and care about the position. Applications were then filtered via a sliding GPA scale, so that students with excellent application scores with lower GPAs still would get the opportunity to interview. After the filter, 54 out of the original 81 applicants received interview offers, with 45 ultimately choosing to complete an interview.

Interviews were conducted with three applicants per group, meeting virtually with the Head uTA for 45 minutes. The group setting reduced time between interviews through answering common questions, and most importantly, provided an opportunity to see how each candidate interacted with their peers. Rather than focusing on content knowledge, we developed an interview rubric similar to [2], which places more emphasis on holistic characteristics, such as teaching and empathy skills over technical skills. Each candidate took turns answering questions from our newly developed question pool, which included "standard" interview questions ("Why do you want to be a TA?"), realistic scenarios ("How would you respond to a student getting overwhelmed and crying during office hours?"), position-specific questions ("What is the largest challenge you'd expect to have as a uTA?"), and overarching theme questions ("What is one action you can take as a TA to foster an inclusive and diverse learning environment?").

Having hired the uTAs who displayed both empathy and competence, we chose an asynchronous training format using the campus learning management system (LMS), Canvas. We explored several training programs, handbooks, and other peer mentor/tutor resources to identify topics [4, 6]. The resulting training was organized into three reading and video-based modules with quizzes. Module 0 includes on-boarding procedures, ethics, FERPA compliance, and the course's expectations for teaching assistants. Module 1 prepares uTAs to fulfill their duties, including use of the LMS and Discord, advice for first-time TAs, and preparation for the first lab. Module 2 provides instruction on student learning, effective grading, community building with students, and creating an inclusive classroom. The "TA Training Course" on our LMS continued to serve as a reliable way to contact our staff and share resources throughout the semester.

With 68 uTAs, management is a significant task; this was accomplished by using two Head uTAs. Englert, the poster presenter, served as the Head uTA for hiring and training during the summer, and shared responsibility for management with another senior uTA. The Head uTA does not work directly with a cohort of students, but instead is responsible for checking in with the uTAs, helping to find coverage in a uTA's absence, address uTAs' questions and concerns, and manage the LMS.

4 EVALUATION

To evaluate the intervention, we compared instructor time and learning outcomes to pre-intervention levels. We found that instructors spent less time lecturing overall, with more time to hold office hours and answers students' questions via email. Despite less mandatory in-class time per week, the post-intervention students outperformed pre-intervention students in final course grades ($p=.01$). In addition, median delay time for uTAs to manually grade assignments was substantially lower, going from 2.5 weeks to six days ($p<.001$).

In a course survey, most students agreed that their uTA made them feel more connected to the course. Finally, we also sought feedback about hiring, training, and the semester as a whole from uTAs, whose impression of the work was very positive.

ACKNOWLEDGMENTS

We would like to thank our incredible team of undergraduate teaching assistants. This study would not have been possible without their hard work and willingness to take on new challenges.

REFERENCES

[1] Adrienne Decker, Phil Ventura, and Christopher Egert. 2006. Through the Looking Glass: Reflections on Using Undergraduate Teaching Assistants in CS1. *SIGCSE Bull.* 38, 1 (March 2006), 46–50. https://doi.org/10.1145/1124706.1121358

[2] Dan Leyzberg, Jérémie Lumbroso, and Christopher Moretti. 2017. Nailing the TA Interview: Using a Rubric to Hire Teaching Assistants. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (Bologna, Italy) *(ITiCSE '17)*. Association for Computing Machinery, New York, NY, USA, 128–133. https://doi.org/10.1145/3059009.3059057

[3] Rosalie P. Lynch and Eric Pappas. 2017. A Model for Teaching Large Classes: Facilitating a "Small Class Feel". *International Journal of Higher Education* 6, 2 (April 2017), 14 pages. https://doi.org/10.5430/ijhe.v6n2p199

[4] Diba Mirza, Phillip T. Conrad, Christian Lloyd, Ziad Matni, and Arthur Gatin. 2019. Undergraduate Teaching Assistants in Computer Science: A Systematic Literature Review. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. Association for Computing Machinery, New York, NY, USA, 31–40. https://doi.org/10.1145/3291279.3339422

[5] Anya Tafliovich, Jennifer Campbell, and Andrew Petersen. 2013. A Student Perspective on Prior Experience in CS1. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 239–244. https://doi.org/10.1145/2445196.2445270

[6] Luther A. Tychonievich. 2017. *Training Course for Teaching Assistants in Computing*. Retrieved October 10, 2022 from https://www.cs.virginia.edu/luther/ta-training

"Their religion is yoga": Understanding the Role of Spirituality for Black Women in Computing

Miracle Etim-Andy
University of Maryland
College Park, MD, USA
metimand@umd.edu

Sheena Erete
University of Maryland
College Park, MD, USA
serete@umd.edu

Yolanda A. Rankin
Emory University
Atlanta, GA, USA
yrankin@emory.edu

ABSTRACT

Research suggests that despite the growing number of Black women who intend to major in computer science, adverse experiences in hostile computing degree programs contribute to their leaving the field as an act of resistance. The Black women who choose to persist utilize various mechanisms to help them navigate hostile environments within computing. One such mechanism is social capital where Black women leverage their social connections with others in the field of computing. As a part of a larger study that explores the role that social capital plays for Black women in computing, we used the Social Capital Community Benchmark Survey (SCCBS) to ask 14 Black women in three virtual focus groups about the social influences that impact their ability to persist in computing. Results reveal that spirituality is a large part of persistence in computing. However, SCCBS privileges certain religious denominations as part of the narrative of what constitutes acceptable religious practices in the United States. Black women criticize the SCCBS as being non-inclusive of the Black experience in the U.S. Furthermore, Black women indicate that spirituality more so than religious affiliations provides strong social connections that affirm their sense of self while providing a community of support.

ACM Reference Format:
Miracle Etim-Andy, Sheena Erete, and Yolanda A. Rankin. 2024. "Their religion is yoga": Understanding the Role of Spirituality for Black Women in Computing. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635594

1 INTRODUCTION

According to the Computing Research Association Taulbee Survey, Black women exhibit the highest level of aspiration and intent to major in Computer Science (CS) when compared to other racial groups of women in the United States [10, 12]. However, they continue to represent a dismal number of those students graduating with bachelor degrees in CS [6, 7]. Research suggests that the lack of retention is a result of exposure to hostile and systemically oppressive environments once they matriculate into computing degree programs [2, 3, 6]. In recent CS education literature, Black women have reported being faced with gendered racism in the form of isolation, belittlement, and negative stereotypes that portray them as incapable of engaging in computing [6, 7, 9, 11].

While some Black women choose to leave the field as an act of resistance, recent research grounded in Black feminist epistemologies, or Black women's ways of knowing, indicate that Black women have cultivated other coping strategies to resist systemic oppression and persist in computing [1, 2]. Though these scholars emphasize mentorship, role models, and allies, we know little about how Black women leverage social capital, as it relates to religion and spirituality, to navigate the computing ecosystem. In this poster, we address the research question: What role does spirituality play in Black women's ability to persist in computing? As a part of a larger study, we repurpose the Social Capital Community Benchmark Survey (SCCBS) to gauge how spirituality relates to Black women's persistence in computing. Conducting a series of virtual focus groups, we engaged 14 Black women in computing in intimate conversations about their religious affiliations and spiritual practices.

2 BACKGROUND

Our work builds on existing literature that emphasizes Black women's role in the Black church as well as spirituality as a coping mechanism for the retention of Black women doctoral students [2, 3]. Faith-based communities and spiritual practices are said to be sacred spaces that allow Black women to fellowship, be authentic and vulnerable, receive mentorship, and be replenished when mentally or emotionally depleted [3]. Thomas et al. [11] found that Black women PhD students are supported by other Black women, who serve as spiritual leaders that help, protect, provide, and watch out for them. Understanding these friendship networks and other associations used to achieve individual or collective objectives is social capital [8]. Prior research suggests that improved social capital leads to improved academic performance, economic success, better health, and increased quality of life [8]. Despite these connections between spirituality and social capital, little research has explored how Black women leverage spirituality, or how spirituality relates to their social capital as it correlates to their improved ability to thrive in computing education. By invoking testimonial authority and sharing direct quotes of Black women in computing, this project offers novel insights into how spirituality is leveraged to help Black women persist, thrive, and succeed in computing.

3 METHODS

Applying standpoint theory [4], we describe our positionality as three Black feminist scholars who have strong interests in the disruption of the dominant sociocultural norms in computing and who acknowledge Black women's lived experiences as valid forms of

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635594

knowledge. We conducted 3 virtual focus groups between June and July 2020 with 14 Black women in different areas of computing, including undergraduate and graduate students, computing faculty, and industry professionals. Participants were recruited using snowball sampling. The sessions resulted in 5 hours and 27 minutes of video recordings. Prior to each session, participants were asked to complete the SCCBS [8]. During each session, we asked participants their opinions on the relevance and effectiveness of the survey questions in measuring the social capital of Black women's experiences in computing. After each session, the research team iteratively modified the survey based on participants' feedback, presenting a modified version of the survey to the following focus group. We analyzed the anonymized transcripts using the Sort and Sift qualitative approach, where we identified and analyzed notable quotes from each virtual focus group to "understand its content, dimensions, and properties, and then [..] assess[ed] what [we ..] learned in order to bridge findings with current conversations in the field and to assess implications" [5, p.2].

4 FINDINGS

Three themes emerged from our preliminary analysis. First, our findings suggest that some Black women in computing identify with spirituality and religion as a practice as well as a tool of resistance and a place of belonging that should be discussed in computing. For instance, Monica said, *"I know people who [..] their religion is yoga, and if they couldn't do yoga they would cease to exist, just not be able to cope emotionally with anything. [..] They do have [..] a strong community."* In this quote, Monica describes the importance of religion and spirituality, referring to it as life and death for some. In the same, Olivia believed that it was appropriate and necessary to ask questions about spirituality when trying to understand ways in which Black women persist in computing. She said, *"I thought it was interesting that spirituality was one of the [survey question] options, because I think that's not often like talked about [..] as far as like, how you belong in computing in general."*

Second, we found that spirituality, not religious affiliation, is a more accurate measure of social capital. Our participants believed that spirituality requires being in community with others—whether it is attending faith-based events (e.g., worship services) or making time to focus on one's individual wellbeing—and that it is important to ask. Jodie, for example, was pleased that the survey question (in an updated version) asked specifically about spirituality, saying, *"I appreciated that you asked that question on spirituality and not religious denomination."* This indicates a belief that spirituality differs from religion in its capability to offer versatility in the spaces and communities Black women can utilize to focus on their spiritual wellbeing and find support. Similarly, Linda suggests that the survey question should further probe into the impact of spirituality by asking, *"Are you a spiritual person? And then how does this affect your friend networks? How does this affect your [computing] work?"*

Lastly, our findings suggest that the SCCBS was inadequate for understanding the nuances of Black women's social capital in the context of how they navigate computing. For example, the SCCBS asked people to identify their religious denominations, listing options such as Catholic, Protestant, etc. However, Black women did not identify with these stratifications. Instead, our participants suggested that the question emphasizes certain religious denominations that originate from Eurocentric culture, while perceiving others as the antithesis of what it means to be an American. Janice says that while she believes that *"there is some value in understanding how people's faith [..] either marginalized them or also provides them social capital"* in computing, she also feels that the SCCBS is inadequate and exclusionary, because *"there's a lot of stuff other than that, right? And it kind of even just disregarded potential religious experiences that didn't feel kind of important to [the] American narrative, right?"* Most participants found the initial questions about spirituality to be divisive rather than inclusive, leading us to edit the questions to explore spirituality rather than formal religious affiliations.

5 DISCUSSION AND CONCLUSIONS

Our results suggest that Black women in computing utilize spirituality as a means of social connection and self-affirmation. Additionally, we found that the SCCBS does not adequately capture the nuanced and complex relationship between Black women's spiritual practices and social capital as it relates to their engagement in computing. Though SCCBS is a standard tool for understanding social capital in many areas, our findings reveal a need to create more inclusive measures to understand social capital. In future work, we will share new methods and survey tools used to understand Black women's experiences in computing.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1937759.

REFERENCES

[1] Patricia Hill Collins. 1989. The Social Construction of Black Feminist Thought. *Signs* 14, 4 (1989), 745–773. http://www.jstor.org/stable/3174683
[2] Patricia Hill Collins. 2000. *Black Feminist Thought.* Routledge, New York NY.
[3] Sharon Fries-Britt and Bridget Turner Kelly. 2023. *Black Women Navigating the Doctoral Journey: Student Peer Support, Mentorship, and Success in the Academy.* Taylor & Francis.
[4] Sandra G. Harding. 2004. The Feminist Standpoint Theory Reader.
[5] Raymond Maietta, Paul Mihas, Kevin Swartout, Jeff Petruzzelli, and Alison B Hamilton. 2021. Sort and Sift, Think and Shift: Let the Data Be Your Guide An Applied Approach to Working With, Learning From, and Privileging Qualitative Data. *Qualitative Report* 26, 6 (2021).
[6] Yolanda A. Rankin and Jakita O. Thomas. 2020. The Intersectional Experiences of Black Women in Computing. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 199–205. https://doi.org/10.1145/3328778.3366873
[7] Yolanda A. Rankin, Jakita O. Thomas, and Sheena Erete. 2021. Black Women Speak: Examining Power, Privilege, and Identity in CS Education. *ACM Trans. Comput. Educ.* 21, 4, Article 26 (oct 2021), 31 pages. https://doi.org/10.1145/3451344
[8] Saguaro Seminar. 2000. *Social capital community benchmark survey.* TNS Intersearch.
[9] Amber Solomon, Dekita Moon, Armisha L. Roberts, and Juan E. Gilbert. 2018. Not Just Black and Not Just a Woman: Black Women Belonging in Computing. In *2018 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. 1–5.
[10] Jakita O. Thomas. 2023. Intersectional Computing—Where It All Began: A Sankofa Story (Part 1). *Interactions* 30, 3 (may 2023), 63–67. https://doi.org/10.1145/3592860
[11] Jakita O. Thomas, Nicole Joseph, Arian Williams, Chan'tel Crum, and Jamika Burge. 2018. Speaking Truth to Power: Exploring the Intersectional Experiences of Black Women in Computing. In *2018 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. 1–8. https://doi.org/10.1109/RESPECT.2018.8491718
[12] Stuart Zweben and Betsy Bizot. 2020. 2019 CRA Taulbee Survey. *Computing Research Newsletter (CRN)* 32, 5 (2020), 3–74.

Enhancing Code Tracing Question Generation with Refined Prompts in Large Language Models

Aysa X. Fan
xuemof2@illinois.edu
University of Illinois at Urbana-Champaign
Champaign, IL, USA

Rully A. Hendrawan
ruhendrawan@pitt.edu
University of Pittsburgh
Pittsburgh, PA, USA

Yang Shi
yshi26@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Qianou Ma
qianouma@cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

ABSTRACT

This study refines Large Language Models (LLMs) prompts to enhance the generation of code tracing questions, where the new expert-guided prompts consider features identified from prior research. Expert evaluations compared new LLM-generated questions against previously preferred ones, revealing improved quality in aspects like complexity and concept coverage. While providing insights into effective question generation and affirming LLMs' potential in educational content creation, the study also contributes an expert-evaluated question dataset to the computing education community. However, generating high-quality reverse tracing questions remains a nuanced challenge, indicating a need for further LLM prompting refinement.

ACM Reference Format:
Aysa X. Fan, Rully A. Hendrawan, Yang Shi, and Qianou Ma. 2024. Enhancing Code Tracing Question Generation with Refined Prompts in Large Language Models. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635624

1 INTRODUCTION

Code tracing is pivotal in introductory programming education, enabling students to mentally simulate code execution and comprehend its logic [4, 6]. However, developing effective tracing questions is a complex task for instructors. Large Language Models (LLMs) like GPT-4 have demonstrated potential in automating the generation of educational content, such as code explanations and unit tests, potentially reducing educators' workload [5, 9]. Previous research indicated that LLM-generated questions could be of comparable quality to those created by humans, though with varied expert preferences, highlighting a need to discern what constitutes a "better" tracing question [1].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635624

This paper introduces a refined prompt, designed to guide GPT-4 in generating high-quality code tracing questions, utilizing identified quality features from existing literature [2]. Through expert evaluation, comparing new questions with those identified as preferable in previous work, our results suggest an enhancement in question quality in several aspects, affirming that expert-guided LLMs can indeed produce valuable code tracing questions for programming education [3, 4].

Our investigation is steered by the following research questions:

- What characteristics define quality in code tracing questions?
- Can expert guided prompts help LLMs to generate high-quality questions?
- What insights can be derived regarding the quality of LLM-generated code tracing questions?

2 METHODS

2.1 Features Identification and Expert-guided Prompt

Informed by literature [1, 7–9], we discerned features for high-quality code tracing questions, categorizing them into "Utilized" and "Observed" during LLM question generation. "Utilized" features, directly used in LLM prompts, include Complexity (questions require higher cognitive demand), Guessability (question answers are easy to guess), Appropriate Difficulty, and Reverse Tracing (question is reverse tracing or not). Meanwhile, "Observed" features, anticipated without explicit prompting due to their code snippet basis, encompass Clarity (questions stated clearly), Code Relevance (questions relevant to the code), Edge Cases (questions cover edge cases), and Multiple Answers (questions allow several solutions). This categorization guided the nuanced crafting of LLM prompts to generate questions of anticipated higher quality.

A refined prompt guided the LLM to generate code tracing questions, aligning with identified quality features:

> *In your role as an education expert in an introductory Java programming course, you are equipped with a deep understanding of Java and teaching methodologies. [...] Remember, your questions should stem from two sources - the initial Requirement and the Code the students have produced. [...] Note that a good tracing question have one or more of the following characteristics:*

- *Is a tracing question*

...

- *May have multiple answers*

Your goal is not just to evaluate their coding skills, but to instill a deep, lasting understanding of Java programming in them. [...] For each tracing question you created, also justify the motivation of the question, and why do you think it's a good tracing question. You will ask many relevant tracing questions given 1. Requirement, 2. Code, as below:
Requirement: {desc}
Code: {code}

Additionally, a "chain of thought"[10] approach was utilized: the LLM drafted explanations for each question, then selected the best ones based on the "good question features". For example:

Table 1: Comparison of Old and New Generated Questions

The Best Old Questions	The Best New Questions
If we want to make a pyramid pattern, what changes are needed in the code?	What is the initial value of the variable 'j' in the inner loop, and how does it change in each iteration?
What should be the values of 'x' and 'y' for the code to output "first case"?	If x = 3 and y = 9, which case will be executed, and why?

2.2 Expert Evaluation

An expert in CS Education, with over 5 years' teaching experience, blindly evaluated new and previously deemed "preferable" tracing questions with criteria: (1) Tracing complexity, (2) Reverse-tracing inclusion, (3) Concept keyword mentions, (4) Difficulty appropriateness (1-5 scale), (5) Corner/edge case presence, (6) Answer guessability, and (7) Preference of the question. We expanded upon evaluation criteria from a previous study [1], adding new metrics to capture additional characteristics of high-quality questions.

3 RESULTS AND DISCUSSION

In the evaluation of old and new LLM-generated questions, notable differences were observed. New questions exhibited a higher average complexity (3.5 vs. 2.5) and prominently featured reverse-tracing (83.33% vs. 0%). They also more frequently mentioned concept keywords (33.33% vs. 16.67%) but were slightly less suitable for introductory courses in terms of difficulty level (4.25 vs. 4.67). While new questions did not target corner/edge cases, they were slightly more guessable on average (1.33 vs. 0.92). Interestingly, a marginally higher percentage of new questions were deemed better by the expert (41.67% vs. 33.33%), indicating the potential efficacy of the refined prompts in generating quality questions, albeit with room for further enhancement in certain aspects. Now go back to the research questions:

RQ1: What defines tracing question quality? Quality in code tracing questions may be relevant to complexity, concept keyword inclusion, appropriate difficulty, minimized guessability, and occasionally targeting corner or edge cases.

RQ2: Can prompts guide LLMs to generate high-quality questions? Yes, refined prompts incorporating identified characteristics have successfully guided LLMs to generate questions of improved quality.

RQ3: Other insights? LLMs can generate good quality questions with apt prompts, but challenges persist in areas like crafting reverse-tracing questions and ensuring optimal difficulty levels.

The data indicates that expert-guided prompts enhanced LLM-generated question complexity and keyword usage, yet maintaining consistent difficulty and generating quality reverse tracing questions persist as challenges. This highlights the intricate task of developing prompts that guide LLMs to produce high-quality and suitably tailored questions for the intended student demographic.

4 FUTURE WORK

We plan to extend our methods to a larger dataset of LLM-generated questions to deepen our understanding of question quality, and also plan to conduct more experts' evaluations. Additionally, we aim to incorporate student evaluations, providing insights into their preferences and enhancing the alignment of generated questions with learner needs in programming education. Furthermore, we will explore the application of our refined prompts across various programming languages and contexts, to assess the adaptability and scalability of our approach in diverse educational settings.

REFERENCES

[1] Aysa Xuemo Fan, Ranran Haoran Zhang, Rui Zhang, and Luc Paquette. 2023. Exploring the Potential of Large Language Models in Generating Code-Tracing Questions for Introductory Programming Courses. (2023).

[2] Mohammed Hassan and Craig Zilles. 2021. Exploring 'reverse-tracing'Questions as a Means of Assessing the Tracing Skill on Computer-based CS 1 Exams. In *Proceedings of the 17th ACM conference on international computing education research*. 115–126.

[3] Mohammed Hassan and Craig Zilles. 2023. On Students' Usage of Tracing for Understanding Code. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 129–136.

[4] Amruth N Kumar. 2013. A study of the influence of code-tracing problems on code-writing skills. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. 183–188.

[5] Juho Leinonen, Paul Denny, Stephen MacNeil, Sami Sarsa, Seth Bernstein, Joanne Kim, Andrew Tran, and Arto Hellas. 2023. Comparing Code Explanations Created by Students and Large Language Models. arXiv:2304.03938 [cs.CY]

[6] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. 2008. Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the fourth international workshop on computing education research*. 101–112.

[7] Seán Russell. 2021. Automatically Generated and Graded Program Tracing Quizzes with Feedback. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 2* (Virtual Event, Germany) *(ITiCSE '21)*. Association for Computing Machinery, New York, NY, USA, 652. https://doi.org/10.1145/3456565.3460054

[8] Seán Russell. 2022. Automated Code Tracing Exercises for CS1. In *Proceedings of 6th Conference on Computing Education Practice* (Durham, United Kingdom) *(CEP '22)*. Association for Computing Machinery, New York, NY, USA, 13–16. https://doi.org/10.1145/3498343.3498347

[9] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1* (Lugano and Virtual Event, Switzerland) *(ICER '22)*. Association for Computing Machinery, New York, NY, USA, 27–43. https://doi.org/10.1145/3501385.3543957

[10] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

Generalizing the CS Course-based Undergraduate Research Experience (CURE)

Amanda S. Fernandez
amanda.fernandez@utsa.edu
University of Texas at San Antonio
San Antonio, Texas, USA

ABSTRACT

Undergraduate research experiences have a demonstrated impact on computer science students, however scaling these experiences with growing enrollments can be difficult. Creating experiences tailored to the diverse research fields within computer science can require the support of a large number of faculty and mentors dedicated to these students. Instead, initiatives for group work with focused research topics have been proposed, implemented through the CURE model where students enroll in a course and collaborate to complete research tasks. This work instead proposes a reconfigured CURE for CS, which will enable students to individually pursue a topic in their subfield of interest through a course focused on fundamental research methods and with the support of peers in special interest groups (SIG). The proposed **CS-CURE** retains the original elements of scientific approach, discovery, iteration, collaboration, and contribution, identified as critical components of a CURE. This course is scheduled to be offered at the University of Texas at San Antonio (UTSA) in Spring 2024.

ACM Reference Format:
Amanda S. Fernandez. 2024. Generalizing the CS Course-based Undergraduate Research Experience (CURE). In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635502

1 MOTIVATION

The undergraduate computer science (CS) population at UTSA has grown by 20% each year for the last several years, growing the department to over 1900 undergraduate students and around 20 faculty engaged in research with 80+ masters and 65+ doctoral students. Despite the growing enrollments, Figure 1 shows estimates in Fall 2023 of undergraduate participation in research to be less than 20 total individuals, including paid RA positions, credit (e.g. independent study), and any other unpaid opportunities. To scale undergraduate research opportunities with the rapidly growing population, we explore existing experiences, propose our own, and identify metrics for evaluation of the new model.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635502

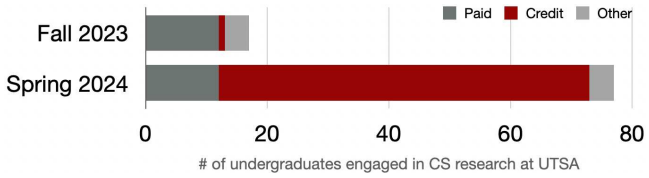


Figure 1: Motivation for this course: comparison of the number of undergraduate CS students at UTSA engaged in research (anticipated) before & after offering CS-CURE.

2 UNDERGRADUATE RESEARCH OPPORTUNITIES AND CURES

With some balanced ratio of faculty to students, courses such as independent study or directed research enable excellent opportunities for one-on-one collaboration, however without a balanced ratio, this model can be difficult for faculty to maintain. Programs such as the Early Research Scholars Program (ERSP) [2] have enabled undergraduate research at scale, modeling a one-year experience with significant support from faculty and mentors [1].

The Course-based Undergraduate Research Experience (CURE) model has been implemented across disciplines [5–8], to enable students to enroll in a course where they collaboratively research a provided topic. The fundamental elements of a CURE model [4] can be summarized as:

- **Scientific Approach**: students learn scientific practices.
- **Discovery**: students gather or analyze novel data.
- **Iteration**: students engage in iterative activities.
- **Collaboration**: research & learning is collaborative.
- **Contribution**: research topic is broadly relevant to the field.

3 A GENERALIZED CS-CURE

The proposed CS-CURE will be implemented as a single semester course where **students will pursue individual research projects** while learning the fundamentals of research methods. The course will adhere to the fundamental CURE elements:

- *Scientific Approach*: course materials will introduce tools & practices, exercised via activities & the research project.
- *Discovery*: research projects based upon literature surveys from recent state-of-the-art in the subfield.
- *Iteration*: in-class activities serve to connect content and iteratively update a semester-long research project with feedback from the instructor and SIG peer groups.
- *Collaboration*: through SIG activities (section 3.1).
- *Contribution*: projects provide a solid foundation for extension in URE/REU, independent study, or graduate programs.

Module	Topic	Activity
1	Research basics	SIG Meeting 0
2	Tools	Guest (AI)
3	Literature review	SIG Meeting 1
4	Research design	Guest (Cloud/IoT)
5	Data analysis	SIG Meeting 2
6	Conference/journal writing	SIG Meeting 3
7	Peer review	Guest (Cybersecurity)
8	Technical presentations	SIG Meeting 4

Table 1: Outline of content in proposed CS-CURE course.

Meeting	Activity
0	Research areas in computer science
1	Identifying research problems
2	Data analysis
3	Abstract writing
4	Peer review activity

Table 2: Student SIG meetings and activities.

3.1 Student Special Interest Groups (SIG)

While students will complete a research project individually, in order to gain the full experience of research in a community, students will also be grouped by research topic. These groups will be aligned with ACM's SIGs to provide further resources, as well as ensure each individual research project is reflective of the interests of the broader community (i.e. the *contribution* element of CUREs). Student SIG meetings will be scheduled as in-class activities, and these will be advertised to all CS faculty at the start of the semester, to encourage engagement if available and interested in promoting undergraduate research in this or future semesters.

3.2 Curriculum

A schedule for the modules of the course is included in Table 1; in our 16 week offering, each module will be 2 weeks. As the course will meet 2 days per week, the first day will include lecture content on the designated topic, and the second day will focus on an activity connecting the topic with the student's individual research project, to include a graded deliverable. Faculty in the department who are searching for undergraduate researchers in the subsequent semesters will provide technical guest lectures as scheduled, in order to introduce their research labs and expose students to diverse areas in CS. Students will meet with their SIGs on a regular basis, with structured activities (outlined in Table 2) with a graded deliverable in order to reinforce the course concepts.

4 EVALUATION

In order to evaluate the success and impact of the proposed CS-CURE (beyond the number of students engaged as in Figure 1), we will follow previous work [3] and build in a series of surveys for student participants. Through these surveys, we aim to strategically address scale & other challenges of this proposed initiative at UTSA.

4.1 Research Questions

At the start of the course, demographic information will be gathered in order to analyze the following research questions across underrepresented groups in CS as well as first-generation students.

- RQ1: Do students who participate in this CS-CURE feel more confident and included in the CS community?
- RQ2: Did the course influence students' desire to continue with research and/or attend graduate school?
- RQ3: Did the experience alter the students' topic of interest?

The following question serves as an example of the questions on belonging and confidence: "How much do you agree or disagree with the following statements?" (assessed on a 5-point Likert scale):

- *I do not have much in common with students in my CS classes.*
- *I feel prepared for a URE/REU.*
- *I am confident that I can make a contribution to my field.*

4.2 Challenges & Discussion

We aim to bolster survey participation by providing the surveys in conjunction with quizzes due for the course - one in the first week of the semester and one in the final week. If successful, CS-CURE will increase undergraduate research at UTSA slightly however it will not scale with the growing population, if offered annually and by a single instructor. Support of one or more graduate teaching assistants would improve the model, as mentoring has been shown to be critical to scaling such experiences [1], however availability may be limited. In addition to scale of the course, scaling research opportunities beyond the course will need to be considered, including encouraging faculty to increase opportunities for URE/REUs, independent study, & undergraduate positions within their labs.

ACKNOWLEDGEMENTS

This project is partially supported by UTSA / Computer Science.

REFERENCES

[1] Christine Alvarado, Alistair Gray, Diba Mirza, and Madeline Tjoa. 2021. The Role of Mentoring in a Dual-Mentored Scalable CS Research Program. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 945–951. https://doi.org/10.1145/3408877.3432364

[2] Christine Alvarado, Joe Hummel, Diba Mirza, Renata Revelo, and Lisa Yan. 2022. Scaling and Adapting a Program for Early Undergraduate Research in Computing. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*. 50–56. https://doi.org/10.1145/3478431.3499336

[3] Christine Alvarado, Sergio Villazon, and Burcin Tamer. 2019. Evaluating a scalable program for undergraduate CS research. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 269–277. https://doi.org/10.1145/3291279.3339406

[4] Lisa Corwin Auchincloss, Sandra L Laursen, Janet L Branchaw, Kevin Eagan, Mark Graham, David I Hanauer, Gwendolyn Lawrie, Colleen M McLinn, Nancy Pelaez, Susan Rowland, et al. 2014. Assessment of course-based undergraduate research experiences: a meeting report.

[5] Lipika Deka, Peri Shereen, and Jeffrey Wand. 2022. Bringing a CURE into a Discrete Mathematics Course and Beyond. *The Mathematics Enthusiast* 19, 3 (2022), 771–784.

[6] Fahmida Hamid. 2020. A Course-Based Undergraduate Research Experience (CURE) in Computer Science: An Experience Report. *J. Comput. Sci. Coll.* 35, 6 (April 2020), 56–65. https://doi.org/10.5555/3417608.3417614

[7] Katherine Izhikevich, Kyeling Ong, and Christine Alvarado. 2022. Exploring Group Dynamics in a Group-Structured Computing Undergraduate Research Experience. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*. 135–148. https://doi.org/10.1145/3501385.3543959

[8] Paula Lauren. 2021. A Study of Course-Based Undergraduate Research Experiences and the Challenges and Opportunities for Computer Science. *Michigan Academician* 47, 3 (2021), 25–25.

SQLearn: Automated SQL Statement Assessment using Structure-based Analysis

Sumukhi Ganesan
Khoury College of Computer Sciences
Northeastern University
San Jose, CA, USA
ganesan.su@northeastern.edu

Tianchou Gong
Khoury College of Computer Sciences
Northeastern University
San Jose, CA, USA
gong.tian@northeastern.edu

Jeongkyu Lee
Khoury College of Computer Sciences
Northeastern University
San Jose, CA, USA
jeo.lee@northeastern.edu

ABSTRACT

In the world of database education, SQL (Structured Query Language) is often the first and crucial step toward developing data analysis and manipulation skills. As the world moves on to data-driven technologies and businesses, the need to educate students in writing accurate and efficient SQL queries has become paramount. Traditional SQL evaluation modes often rely on limited, subjective, and labor-intensive manual grading, which impedes the integration of practical assignments into the curriculum. This poster introduces SQLearn, an innovative automated assessment tool for SQL education. We aim to build a comprehensive platform that addresses submission, evaluation, and review needs amongst students and educators. Here we highlight our assessment approach which breaks down student-submitted queries into Abstract Syntax Trees (AST) and uses cosine similarity to evaluate them. Experimental results show that the proposed approach is effective, not only in binary grading of queries but also in assigning partial grades. The tool also offers an interactive platform to submit and receive feedback, enabling students to refine their SQL skills and gain insights into query structure and optimization. By automating the assessment process, educators can focus on refining the curriculum and channel more time into instruction and research.

ACM Reference format:

Sumukhi Ganesan, Tianchou Gong and Jeongkyu Lee. 2024. SQLearn: Automated SQL Statement Assessment using Structure-based Analysis. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635607

1 INTRODUCTION

Traditional challenges in the manual evaluation of SQL statements have hindered the seamless integration of practical assignments into the database curriculum. For instructors, manual grading is time-consuming and is not scalable enough to meet the increasing student demands. For students, manual assessment is often subjective and the delay in feedback retards the steady refinement of their query-writing skills [1].

Auto-graders offer a promising solution to these challenges in education by reducing manual effort and evaluation delays. While they have been successful with other programming languages before, SQL assessment presents a unique set of challenges due to their diverse structural nature of SQL statements. We propose SQLearn, an automated SQL statement assessment tool, leverages the inherent structure of SQL statements to analyze and evaluate them. SQLearn compares the student submissions to a preset reference query, provided by the instructor, using structure-based analysis. This approach transcends surface-level syntactic correctness and delves into the logical intent of the statements providing students with fair and timely feedback. This project also aims to build an end-to-end platform for facilitating practical SQL assignments including assignment design, submission, grading, and feedback.

2 RELATED WORK

The landscape of automated SQL assessment tools is diverse and robust [2]. Categorized into dynamic and static analysis approaches, these tools play a crucial role in grading SQL assignments. Dynamic analysis involves executing student queries and comparing results with those of reference solutions, often yielding binary correctness decisions. While popular, this approach faces challenges such as sensitivity to result set differences, susceptibility to incorrect evaluations in specific database contexts, and a lack of insightful feedback on incorrect solutions. In contrast, static analysis compares SQL statements without execution, focusing on query structure to estimate correctness. Techniques like canonicalization and Abstract Syntax Trees (ASTs) are employed for query representation. Some systems combine dynamic and static approaches for a comprehensive evaluation [3].

Throughout this review, two central challenges in automated SQL assessment emerge – the complex task of establishing query equivalence due to SQL's undecidability, where syntactically different queries can be semantically identical, and the need to award fair partial grades for submissions close to, but not identical to, the reference query. SQLearn aims to provide a novel structure-based approach that addresses the challenges identified and to offer a more comprehensive and insightful tool for automated SQL assessment.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635607

3 METHODS

At the core of SQLearn's assessment method, is the Abstract Syntax Tree (AST) which is introduced to capture the distinct structure of SQL queries, defined by keywords like "SELECT," "FROM", "WHERE," and "JOIN." The AST, a product of parsing and optimization by the optimizer, breaks down queries into a tree-like structure, representing their hierarchical and semantic relationships. Utilizing *sqlparse* for tokenization and parsing, the syntactic structure of SQL queries is transformed into ASTs. To facilitate similarity calculations between student and reference queries, a preorder traversal method is employed to systematically explore each node of the AST, creating an ordered representation translated into a sequence using *CountVectorizer*. This representation method is pivotal for SQLearn's objective, requiring similarity between reference and student queries for effective analysis.

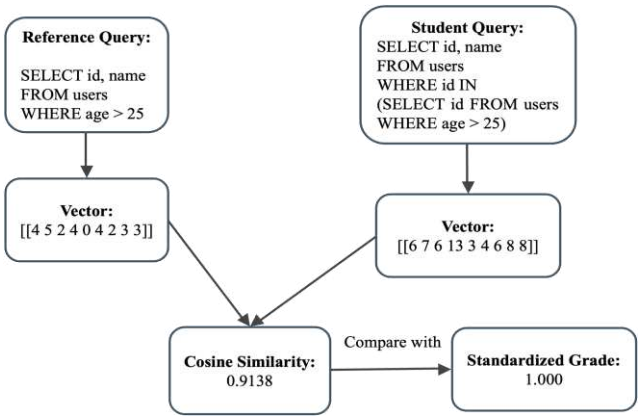


Figure 1: An Example of SQLearn Query Comparison

Feature extraction is foundational for cosine similarity calculation in SQLearn, transforming SQL queries into meaningful characteristics for effective attribute analysis. The cosine similarity measure, S (A, B), gauges the angle between vectors A and B. SQLearn determines the similarity between a reference and student query using this function. As shown in Figure 1, this assesses the cosine angle between vectors, resulting in a 0 to 1 similarity score. SQLearn efficiently evaluates student queries against references. This methodology is integrated into a MERN stack platform for assignment handling and evaluation.

4 RESULTS AND DISCUSSION

To evaluate the effectiveness of the proposed assessment method, two distinct Query Datasets (QDs) containing pairs of student-reference queries and corresponding manual grades are used. The first query dataset (QD1) is created by our team to test the system's robustness covering several possible syntactic and semantic variations. The second query dataset (QD2) was obtained from [3] and acts as our benchmark dataset. For the scope of this study, the query pairs are broadly labeled into three categories - "*correct*", "*incorrect*" and "*partially correct*" by applying pre-determined thresholds on the available manual grades. Table 1 presents average similarity score generated by SQLearn for each category, along with the corresponding average manual grade (standardized to the range of 0 to 1). A closer alignment between the average similarity score and normalized manual grade indicates desired performance.

Table 1: Average Similarity Scores vs Standardized Grades

Dataset	Similarity Scores (Ground Truth)		
	correct	*partially correct*	*incorrect*
QD1	0.90 (1.00)	0.82 (0.86)	0.10 (0.00)
QD2	0.86 (1.00)	0.73 (0.74)	0.62 (0.32)

To classify query pairs based on similarity scores, we determine thresholds that maximize the area under the precision-recall curve for each category. After applying these thresholds, SQLearn's classification accuracy is gauged by comparing results with manual grade classes. Table 2 displays the accuracy achieved for each dataset by category. The results confirm the method's effectiveness in identifying "*correct*" and "*partially correct*" queries. The subjective nature of manual grades and categorization thresholds significantly influence the method's accuracy.

Table 2: SQLearn's Accuracy Rate

Dataset	Accuracy Rate (%)		
	correct	*partially correct*	*incorrect*
QD1	80.0	67.0	100.0
QD2	77.0	62.0	12.0

5 CONCLUSION

In this poster, we introduced SQLearn, a comprehensive automated assessment tool for SQL statements. SQLearn leverages AST and cosine similarity to evaluate student submissions. Our experimental results on query datasets demonstrate its effectiveness and efficiency in categorizing queries as '*correct*', '*partially correct*', or '*incorrect*'. By breaking down SQL queries into their structural components, SQLearn provides a fair and timely assessment, addressing the challenges of traditional manual grading methods. Future work includes refining the assessment approach for better performance and improving the overall tool based on student and educator feedback.

REFERENCES

[1] Samiha Marwan, et. al., 2020. Adaptive Immediate Feedback Can Improve Novice Programming Engagement and Intention to Persist in Computer Science. In Proceedings of ICER '20. ACM, New York, NY, USA, 194–203. https://doi.org/10.1145/3372782.3406264

[2] Nayak, S. et.al., 2022, January. Review of Automated Assessment Tools for grading student SQL queries. In 2022 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-4). IEEE. doi:101109/ICCCI54379.2022.9740799

[3] Wang, J., Zhao, Y., Tang, Z., & Xing, Z. (2020). Combining dynamic and static analysis for automated grading SQL statements. J Netw Intell, 5(4), 179-190.

Using Survival Analysis to Model Students' Patience in Online Office Hour Queues

Zhikai Gao
zgao9@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Adam Gaweda
agaweda@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Collin Lynch
cflynch@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Sarah Heckman
sarah_heckman@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Damilola Babalola
djbabalo@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Gabriel Silva de Oliveira
gsilvad@ncsu.edu
North Carolina State University
Raleigh, NC, USA

ABSTRACT

Promptly and properly addressing students' help requests during office hours is a critical challenge for large CS courses. With a large number of help requests, the queue gets longer and students have to endure long wait times. To address this problem, we try to quantify students' patience in the queue through survival analysis. Our results show that half of the students are willing to stay in the queue after waiting for 142.5 minutes. Moreover, we find that female students, morning requests, returning students, and requests about test failures are more likely to stay in the queue for a longer time.

ACM Reference Format:
Zhikai Gao, Adam Gaweda, Collin Lynch, Sarah Heckman, Damilola Babalola, and Gabriel Silva de Oliveira. 2024. Using Survival Analysis to Model Students' Patience in Online Office Hour Queues . In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635517

1 INTRODUCTION

Quickly and accurately addressing help requests during office hours (OH) is important for large CS courses [3, 4, 8]. However, those requests often come in large batches. With limited time and resources, it is impossible to provide immediate responses for all students, and thus many students may have a long wait in the queue. These long wait times may cause some students to exhaust their patience, resulting in their leaving the queue and thus not receiving assistance with their issues. As CS continues to grow in enrollment, our instructors are increasingly hearing this negative experience as a major complaint from students. Thus to reduce such cases, it is crucial to understand and analyze how much time students are willing to stay in queue, why they are help-seeking, and who is willing to stay longer. Therefore, in this work we aim to answer the following research questions:

- RQ1: How can we use survival analysis to estimate students' patience in the office hours queue?
- RQ2: Do certain factors influence a student's patience while waiting in the queue?

Researchers have rarely sought to quantify students' patience in the office hours queue; thus we aim to fill this gap in this work. In our prior work[5], we used the distribution of student cancellations to model students' patience. However, this approach discarded wait time data on completed requests, which is the majority of cases. To address our prior limitations, we built a more robust and advanced model by using survival analysis.

Survival analysis[6] is a robust statistical approach that focuses on comprehending the temporal aspect of specific events. It tackles questions about the proportion of a population expected to endure, or survive, past a specific point and whether multiple factors impact the probability of survival. Unlike standard regression methods, survival analysis accounts for the non-normal distribution of event times and censoring in data, which is common in survival studies. While course office hours are often fixed, when students attend office hours mirrors these aspects. Survival analysis also has been widely applied to study user or customer behaviors in queuing systems. For instance, Aksin et al.[1] investigated the relationship between observed queue characteristics and the probability of reneging. They found that when customers observed a quickly moving queue at the beginning, they would be more likely to stay in the queue and exhibit more patience. Since the online office hours process is also a typical queuing system, we believe that survival analysis is a suitable approach.

2 DATASET

The data used in this study comes from the Fall 2020 offering (F20) of a CS2 course taught at an R1, public university in the south-eastern United States (N=303). The course uses Java as a programming language and covers object-oriented programming, basic data structures, finite state machines, recursion, and software engineering concepts. Due to the COVID-19 pandemic, the course was taught completely online during F20. Throughout the semester, students complete three guided projects, two large projects, and 12 lab sessions. The instructor uses GitHub to manage students' code submissions for these activities. Each submission's commits are also automatically compiled and tested using the Jenkins continuous

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635517

integration tool. This tool then provides feedback to students. Students may submit their code an unlimited number of times, using the feedback to improve the correctness of their code solutions.

MyDigitalHand (MDH) [7] is a ticketing system for help requests and is used in the course to manage office hours. Students queue through MDH by virtually raising their hand through a form, which includes a description of their question to the teaching staff. Requests can only be raised when an office hour shift is started. When a member of the teaching staff is available to help the next student, they can select a student from the queue and inform them to enter the Zoom meeting and start the interaction. After the interaction ended, the teacher would go back to MDH to close the request ticket in the queue and continue with the next request. Office hours were held fully online by 16 teaching staff during F20, hosting office hours for at least two hours per week.

In this study, we collected the description, request time, start time, cancel time, and the resolved time for each office hour's help request through MDH. We define the time difference between request time and start time as *wait time*. We removed all requests which have an interaction time of less than 10 seconds since communication of less than 10 seconds is highly unlikely to resolve any issue. We also removed requests made during the night but were canceled or resolved before the next morning. Finally, based on the description, three experts manually categorized the requests about the large projects as Implementation, Test Failure, Debugging, or Other.

3 METHODOLOGY

We applied survival analysis on students' wait time to model how much time students are willing to wait in the office hours queue. When we observe a student has waited t minutes before canceling the request, then we believe their max tolerable waiting time for that request is t minutes. On the other hand, if a student waited t minutes before starting the interaction, then their tolerable wait time is at least t minutes, which can be viewed as a right censored observation in survival analysis. So we use the Kaplan-Meir algorithm to calculate the survival function $S(t)$; such function represents students' tolerable waiting time in a queue. In other words, $S(t)$ equals the probability that a student can still tolerate and wait in the queue after waiting t minutes in the queue.

We then divided students' requests into different groups based on their gender, request time of day (Morning: 8 am-12 pm, Afternoon: 12 pm-5 pm, and Evening: 5 pm-9 pm), wait commits (whether the student made any commit while waiting), previous usage (fist timer or returning student), and request type (Implementation, Debugging, Test Failure, or other). Then, we calculated the survival function for each group of requests separately. Finally, we applied log-rank tests[2] to see if the survival functions are significantly different between student groups.

4 RESULTS AND CONCLUSIONS

We found in the resulting survival function that the median tolerable waiting time is 142.5 minutes ($S(t) = 0.5$, when $t = 142.5$), meaning half of the students are willing to wait for 142.5 minutes in the queue. In addition, 80% of the students are willing to wait for 33.9 minutes. Students are surprisingly patient in queues, and we believe that the online format made that high toleration possible.

	Count	Median Patience	p-value
Male requests	1693	133.2 mins	<0.01
Female request	1203	158.1 mins	
Morning	1536	194.2 mins	<0.01
Afternoon	1417	136.1 mins	
Evening	223	52.1 mins	
Has wait commit	1142	138.9 mins	0.67
No wait commit	2215	144.7 mins	
First timer	203	84.7 mins	<0.01
Returning students	2973	145.4 mins	
Implementation	334	132.9 mins	0.35
Debugging	392	139.2 mins	
Test Failure	260	150.7 mins	
Other	447	145.7 mins	

Table 1: Correlation between students' patience and the characteristics of their requests.

When requesting help online, students are still free to do other things without any physical limitation, and they do not suffer from the experience of being stuck in a long physical queue.

Table 1 shows how different factors related to students' patience. We found that female students, morning requests, returning students, and requests about test failures were significantly more likely to stay in the queue for a longer time ($p < 0.05$). The high patience of female students might be caused by their low confidence in problem-solving skills based on previous research[4]. Also, returning students might gradually get used to the long wait; hence their observed patience increases. Even though we found those students/requests are more likely to endure the long wait, teachers shouldn't exploit that feature and help other students first. In the future, we will build an ML model to predict each request's estimated tolerable wait time based on our results; then based on this prediction, we can monitor the queue status and warn the teacher when the queue gets too long and most students will probably run out of patience before they can receive help.

REFERENCES

[1] O Zeynep Aksin, Busra Gencer, and Evrim Didem Gunes. 2022. How observed queue length and service times drive queue behavior in the lab. *Available at SSRN 3387077* (2022).

[2] J Martin Bland and Douglas G Altman. 2004. The logrank test. *Bmj* 328, 7447 (2004), 1073.

[3] Zhikai Gao, Bradley Erickson, Yiqiao Xu, Collin Lynch, Sarah Heckman, and Tiffany Barnes. 2022. You asked, now what? Modeling Students' Help-Seeking and Coding actions from Request to Resolution. *Journal of Educational Data Mining* 14, 3 (Dec. 2022), 109–131.

[4] Zhikai Gao, Sarah Heckman, and Collin Lynch. 2022. Who Uses Office Hours? A Comparison of In-Person and Virtual Office Hours Utilization. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1* (Providence, RI, USA) *(SIGCSE 2022)*. ACM, New York, NY, USA, 300–306.

[5] Zhikai Gao, Collin Lynch, and Sarah Heckman. 2023. Too long to wait and not much to do: Modeling student behaviors while waiting for help in online office hours.. In *Proceedings of the 7th Educational Data Mining in Computer Science Education (CSEDM) Workshop*.

[6] David Machin, Yin Bun Cheung, and Mahesh Parmar. 2006. *Survival analysis: a practical approach*. John Wiley & Sons.

[7] Aaron J. Smith, Kristy Elizabeth Boyer, Jeffrey Forbes, Sarah Heckman, and Ketan Mayer-Patel. 2017. My Digital Hand: A Tool for Scaling Up One-to-One Peer Teaching in Support of Computer Science Learning. In *Proceedings of the 2017 ACM Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. ACM, New York, NY, USA, 549–554.

[8] Matthew Zahn and Sarah Heckman. 2023. Observations on Student Help-Seeking Behaviors in Introductory Computer Science Courses. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2* (Toronto ON, Canada) *(SIGCSE 2023)*. ACM, New York, NY, USA, 1380.

Restorying with AI Art among Latinx Elementary Students

Leiny Garcia
University of California, Irvine
Irvine, CA, USA
leinyg@uci.edu

Santiago Ojeda-Ramirez
University of California, Irvine
Irvine, CA, USA
sojedara@uci.edu

Mark Warschauer
University of California, Irvine
Irvine, CA, USA
markw@uci.edu

ABSTRACT

The Latinx community is underrepresented in tech-oriented fields, which aligns with the lack of culturally relevant learning experiences in CS for Latinx youth, hindering their ability to conceptualize technology as a tool for transformation and vehicle for cultural expression. This study takes on a restorying approach at the elementary level, where 9 fourth-grade students engaged in focus group discussions over three days to generate prompts for a generative AI art. Through the lens of restorying, the prompts had students conceptualize a future with a focus on their Mexican-American heritage, local community, and technology. The study revealed that students associated their heritage with symbolic representations such as food and music, and characterized the community as a commercialized space while also emphasizing locations conducive to family-oriented activities. As a result, technology in community spaces was associated with consumerism. However, when envisioning a futuristic, transformed community, they made deeper connections between the role of technology in the community, making intricate connections between community improvements and technology-based solutions. This underscores the need for computing education to dedicate time for young learners to reflect on the role technology has on their current culture and community to make deeper connections.

KEYWORDS

restorying, heritage, elementary school, latinx, AI literacy

ACM Reference Format:
Leiny Garcia, Santiago Ojeda-Ramirez, and Mark Warschauer. 2024. Restorying with AI Art among Latinx Elementary Students. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635618

1 INTRODUCTION

The Latinx community stands as the second-largest racial and ethnic group in the US [4]. Yet, there's a continued disparity in participation in tech-oriented fields [1]. The participation gap has been attributed to a lack of culturally relevant learning experiences that allow historically marginalized youth to bring their communities' practices, traditions, and values as they learn about technology. Situated approaches to computational thinking (CT) prioritize students' "personal creative expression and social engagement as a pathway in becoming computationally fluent building on youth interest in digital media and production" [2]. For instance, *restorying* [7], a practice where minoritized youth re-elaborate narratives about their communities to allow for critical reflection on dominant narratives in computer science (CS) [6]. Such approaches have yet to be explored at the elementary level due to mixed findings on the early development of deficit perceptions (i.e., gendered or racial beliefs) [3].

Aiming to expand the implications of restorying approaches, we designed a three-day restorying activity for elementary students. The activities centered their lived experiences, communities, and heritage in relation to computing and technology, encouraging them to speculate about a tech-oriented future for their communities using a text-to-image Generative AI (Dall-e). We explore how these elementary-aged students establish connections between community, heritage, and technology; guiding ourselves by the following research questions: What narratives do Latinx elementary students possess regarding their heritage and communities in relation to computing technology? How do elementary-aged Latinx students construct prompts about alternate futures using text-to-image Generative AI post a CT curriculum?

2 METHODS

The research conducted involved a focus group comprising fourth-grade students enrolled in a dual immersion school in Southern California. Nine students, including six females and three males, were selected to participate in this study. Students were selected by educators based on student interest in computing, willingness to engage in conversation, and availability. Participants were part of a broader initiative to undergo a year-long CT curriculum and this study was conducted at the end of the school year. Therefore, all students have had exposure to the fundamental concepts and practices related to CT. Additionally, the school for this study is predominately Latinx (97%) and our participants are of Mexican-American heritage.

The activity unfolded over three days, with each session lasting 60 minutes. The first day of the workshop was focused on eliciting student perceptions of their current realities through prompts such as *What does being Latinx mean to you?*, *What places in (community) are meaningful to you?*, *Where do you see technology in (community?)*. The second day focused on the creation of their stories and prompts for the generative AI art. Sample prompts that guided a speculative future approach included *What about (Location) is your character going to help improve?*, *Think about a technology that does not exist yet and can help your character.* On the final day, students shared the images created by the AI art generator from their prompts and pondered the correlation between their prompts and the resulting

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635618

images. Data collection methods included audio recordings of discussions, tangible and digital artifacts created by participants, and researcher's memos. Data analysis involved a qualitative inductive analysis of the transcribed discussions and written text in artifacts.

3 RESULTS

3.1 Heritage

When students were prompted to articulate the essence of "being" Latino and Latina, the initial reaction was a confusing statement of, "What?". Once a student shouted "Tacos!", it led students to brainstorm a plethora of similar cultural artifacts and symbols. Their associations with Latinidad primarily clustered around four key themes: Food, Music, Media, and Symbolic Heritage, correlating with themes historically noted to be incorporated by elementary students into their CS projects [5]. Under the umbrella of food, students cited traditional dishes, and in the realm of music, their focus shifted from current Latin music singers to traditional sounds like Mariachi and Folklorico. While these themes predominated, students also referenced contemporary elements such as telenovelas and symbolic representations like the Mexican flag. Interestingly, the discussions geared toward visible aspects of their heritage, notably omitting considerations of values, languages, or beliefs. Moreover, students directly associated culture with their Mexican-American heritage, defining it as "people from Mexico and the culture of Mexico and what's exciting." This definition along with how they defined Latinx heritage sheds light on their evolving perceptions of a cultural-based identity.

3.2 Community and Technology

We found students' concept of community was intimately linked with locations associated with familial and social bonds, such as parks, schools, and family homes. Additionally, students expressed joy in places offering food and entertainment, exemplified by enthusiastic mentions of popular business chains. When queried about their observations of technology in their community, students predominantly cited retail establishments. While less mentioned, children also acknowledged technology within their immediate surroundings, including school, libraries, and their homes. Their initial perceptions of technology leaned heavily toward human-centered devices, such as iPads, tablets, televisions, gaming consoles, and popular social apps like TikTok.

When prompted to envision changes in their community, students showcased a more nuanced and expansive perspective on the role of technology. Their responses reflected a desire for improvements in areas such as pollution, homelessness, and safety. Although students couldn't pinpoint existing technologies addressing these concerns, they quickly resorted to imagined futures and discussed potential solutions. One student passionately stated, "Robots can pick up trash around our cities and the ocean," exemplifying how they can imagine the role that technology can have on pollution-related issues. Moreover, the discussions revealed a growing awareness among students regarding the environmental consequences of technology. A student astutely pointed out, "not really too because robots go on electricity and electricity burns and then creates fog that creates pollution." This remark highlighted their ability to evaluate the impact of technology through an imagined world, indicating a developing understanding of the intricate relationship between technological solutions and community.

3.3 Alternate futures through AI Art

In terms of culture, students demonstrated their ability to envision technology enhance cultural practices that shape meaning, such as "A latino family in California using future technologies like computers, electronics and robots to experiment and make food like tacos. They should be cooking with robots." Yet, the majority of students depicted their characters engaging in cultural practices devoid of any technological influence. Simultaneously, while framing their characters within community spaces through a consumerist lens, they ingeniously integrated technology as a means of fostering community advancement. Such as the following prompt: "[Character] goes to the mall and uses a hologram-watch to fight climate change, with Cholete the dog, playing traditional Mexican corrido music."

4 DISCUSSION AND FUTURE WORK

Ensuring the inclusivity of historically marginalized groups in CS and AI necessitates empowering students to recognize technology not just as technical devices but as instruments that can transform spaces and shape their evolving identities and realities. Our study delves into crucial opportunities for deeper restorying discussions for elementary-aged students. Findings surrounding narratives on culture, community, and technology reveal that students need opportunities to reflect on their cultural identities and community spaces for deeper connections on the role of technology in those realms. As a result, a consumerist lens dominated the conversations on the role that technology has in their immediate realities. Yet, elementary students can make deeper connections when discussing imagined futures of their community and technology, creating prompts that embed technology into cultural practices (i.e., cooking) and community needs. Future iterations of this design would include longer discussions and spaces of reflection about historical practices and current discourses about communities with students.

REFERENCES

[1] Glenda M Flores. 2011. Latino/as in the hard sciences: Increasing Latina/o participation in science, technology, engineering and math (STEM) related fields. *Latino Studies* 9 (2011), 327–335.

[2] Yasmin Kafai, Chris Proctor, and Debora Lui. 2020. From theory bias to theory dialogue: embracing cognitive, situated, and critical framings of computational thinking in K-12 CS education. *ACM Inroads* 11, 1 (2020), 44–53.

[3] Allison Master. 2021. Gender stereotypes influence children's STEM motivation. *Child Development Perspectives* 15, 3 (2021), 203–210.

[4] Luis Noe-Bustamante, Mark Hugo Lopez, and Jens Manuel Krogstad. 2020. US Hispanic population surpassed 60 million in 2019, but growth has slowed. (2020).

[5] Santiago Ojeda-Ramirez, Jennifer Tsan, Donna Eatinger, Sharin Jacob, Dana Saito-Stehberger, Diana Franklin, and Mark Warschauer. 2023. Describing Elementary Students' Spheres of Influence in Scratch'About Me'Projects. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 333–339.

[6] Mia S Shaw, James Joshua Coleman, Ebony Elizabeth Thomas, and Yasmin B Kafai. 2023. Restorying a Black girl's future: Using womanist storytelling methodologies to reimagine dominant narratives in computing education. *Journal of the Learning Sciences* 32, 1 (2023), 52–75.

[7] Ebony Elizabeth Thomas and Amy Stornaiuolo. 2016. Restorying the self: Bending toward textual justice. *Harvard Educational Review* 86, 3 (2016), 313–338.

Augmenting Computer Science Undergraduate TA Training Programs to Benefit Recruitment, Refinement, and Retention

Julie V Gawenda
University of Illinois Chicago
Chicago, Illinois, USA
jvgawenda@gmail.com

Adam T Koehler
University of Illinois Chicago
Chicago, Illinois, USA
akoehler@uic.edu

Agne Nakvosaite
University of Illinois Chicago
Chicago, Illinois, USA
agnenakvo@gmail.com

ABSTRACT

Undergraduate Teaching Assistants (TA) can be a dynamic resource for any Computer Science (CS) program. However, increasing the TA pool size is often the primary method explored when investigating department TA improvements, but budgetary restrictions frequently negate that possibility. Additionally, undergraduate students applying to TA for classes do not typically come from an educational background or have teaching foundations. With the expected natural turnover of graduation and other life or academic factors, TA retention is always an issue, but our institution has encountered abnormally low TA retention (34% 2021 to 2022), indicating other factors may be contributing to attrition. We investigate alternative ways to improve TA skill sets to benefit the department in TA recruitment, refinement, and retention. Our modules aim to directly impact TA refinement while indirectly impacting TA recruitment and retention. Building upon established research and drawing inspiration from existing programs at several North American universities, we cultivated four interactive modules for undergraduate TA training programs in CS that aim to improve TA skill sets and self-efficacy. We provide rationale and details for each module alongside an example implementation to ease external adoption of ours or similar modules targeting localized needs.

ACM Reference Format:
Julie V Gawenda, Adam T Koehler, and Agne Nakvosaite. 2024. Augmenting Computer Science Undergraduate TA Training Programs to Benefit Recruitment, Refinement, and Retention. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635625

1 PROBLEM & MOTIVATION

We define the three pillars of effective computer science (CS) undergraduate teaching assistant (TA) training programs as recruitment, refinement, and retention. The TA recruitment and retention process generally is 1) recruit from the student population, 2) hire, train, and hope TAs continue in future terms without additional motivation, and 3) periodically recruit again to handle expected attrition. Even though the necessary skills can vary by course, every CS TA should be proficient in leading exercises, guiding students during development without solutions, and exercising perceptiveness during office hours. Many universities provide training or guidance through programs to enhance and refine the skills of new TAs [4–9], but they often lack an interactive element simulating TA experiences. Our contributions fill this deficit using interaction and engagement. Additionally, with TA refinement, we aim to replicate the grad TA self-efficacy improvements of DeChenne et al. [3].

Analyzing our university's undergraduate CS TA appointments over five terms exposed a high TA turnover rate, including a 66% turnover rate from 2021 to 2022. While some attrition is expected, such as TA graduation, a high turnover rate can lead to the disorganization of the TAs in courses and a failure to pass on knowledge from one TA cohort to another. Additionally, inadequate training may increase the churn rate. Multiple semesters of experience is an effective pathway to TAs with the skills to handle student problems [8]. However, new TAs have no experience and may be undertrained and overwhelmed. Our modules focus on the "refinement" pillar, as many undergraduate TAs do not come from an educational background. We build upon existing research [6, 9] to prepare TAs for common challenges and transform from a peer to a teacher.

Several programs emphasize TA "refinement" for undergraduate CS TAs. The University of Toronto uses a team-teaching program to encourage community, and participants felt teaching gets less "intimidating" over time as TAs gain additional practice [8]. In contrast, the University of Calgary implements a TA mentorship program between graduate and undergraduate TAs where the graduate TAs provide training, advice, and shadowing [9]. At the University of California, Irvine, a one-quarter seminar is required for new TAs, and they learn about instruction, department policies, and unique teaching situations [6]. Our seminar implementation for new TAs builds upon established practices and incorporates our modules that use facilitated experiences and peer feedback to foster community.

Promoting community can increase TA retention. University of Toronto researchers found that tightly-knit communities developed after eleven mandatory training sessions [4]. Whereas training at the University of Illinois Urbana-Champaign began by empowering TAs with the aim of a downstream effect—training the TAs to promote their students' intrinsic motivation to learn [5]. Patitsas noted that a student's "Eureka" moments were often a TA's favorite part of TA-ing [7]. The positives of established research demonstrate that a training program can increase TA and student retention.

2 MODULES & DESCRIPTIONS

Training programs should refine TA skill sets and promote TA retention. Our four modules (previewed in Table 1) facilitate this through TA interactions and encouraging friendliness and community. Our modules are designed to work in conjunction with any

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635625

Table 1: Overview of Interactive TA Training Modules.

	Absorb & Disseminate	Consolidate & Elaborate	Pseudocode & Drawing Diagrams	Confront & Diffuse
Purpose	Simulate classroom and office hours teaching or Q&A interactions.	Simulate interactions that require teaching a concept in a short period.	Reinforce problem breakdown and rephrasing techniques.	Understand the student's perspective and learn to diffuse confrontation.
Skills	Adaptability, Quick Thinking	Creativity, Communication	Problem-solving, Visualization	Conflict Resolution, Active Listening

required campus-specific curriculum. A typical class will contain an icebreaker to develop community and peer connections, a lecture on responsibilities, a pre-module quiz, module implementation, a post-module quiz, and peer feedback. Our modules use simulated experiences, a process well established in the medical community [2]. Including assessment allows fluid data-based program alterations.

Absorb & Disseminate. During office hours and class, students may approach TAs with exercises that the TA may have limited prior knowledge of. These encounters require adaptability and quick thinking to teach the foundations of a concept without a solution. Moreover, the ability to absorb complex information rapidly is crucial. TAs are often given advanced concepts or computational algorithms and have a short time frame to understand them. This skill goes beyond memorization, involving comprehension, connecting ideas, and distilling complex concepts into digestible bits for students. For our module's implementation, an advanced concept, such as a computational algorithm, is given to the TAs. After ten minutes of "absorption," the TAs "disseminate" the information by reiterating the concept in their own words and presenting it to the other TAs for five minutes. This approach involves unfamiliar concepts, a tight time frame, and rapid dissemination of information.

Consolidate & Elaborate. In student encounters, TAs must quickly gauge a student's comprehension level, adapt explanations based on different learning styles, and foster a supportive learning environment where each student feels comfortable seeking clarification. Tackling all these items demands creativity, problem-solving skills, and a deep understanding of the concept's versatility. With this module, a group of three TAs start by each choosing a programming language and a basic CS concept, such as loops or functions. After this, each TA will "elaborate" on the concept to the other TAs over five minutes. TAs use any available materials, such as whiteboards or paper, to deepen understanding. A give-and-take setup allows TAs to engage the presenter with questions while also experiencing varying teaching styles. Finally, peer feedback is given before presenting TAs are switched.

Pseudocode & Drawing Diagrams. Writing plain language descriptions or drawing out problems are two essential problem-solving techniques. However, students often need clarification on when to deploy each strategy. TAs can further a student's deployment understanding by applying the techniques when helping a student progress with their actual question. As such, TAs must possess a strong command of programming concepts and the ability to simplify complex procedures into understandable, logical sequences. TAs adept in these aspects can then use visual cues to elucidate the problem-solving process, aiding students who learn better through visual aids such as diagrams. In the implementation of this module, TAs receive a coding problem. Then, in fifteen minutes, the TAs develop two step-wise solution pathway presentations, one in plain language and the other using diagrams. This module reinforces different pathways to accommodate diverse learning preferences.

Confront & Diffuse. Office hours and discussion areas can become heated. Students may react poorly to an assignment, feedback, or grades. TAs must learn the art of using appropriate language and body language to diffuse confrontations. This module develops skills for active listening, empathy, and the ability to respond thoughtfully without escalating the situation. In the module's implementation, TAs roleplay as a TA or a student. The "student" is given a failed assignment and develops defensive arguments to "confront" the "TA." The "TA" must "diffuse" the situation and tactfully counter each argument. TAs learn to anticipate and understand the emotional reactions and arguments in various student encounters through roleplay. Roleplaying as a student helps foster empathy for students and builds experience for TAs to steer future encounters toward constructive solutions. Ultimately, TAs learn to manage conflict, communicate effectively, and resolve issues diplomatically.

3 CONCLUSION

Our primary contribution is the four modules and their descriptions, allowing other universities to adopt or adapt our modules directly. Implementing interactive modules within the undergraduate TA training course directly impacts the refinement of CS TAs, and indirectly increases TA retention, thereby reducing the need for recruitment. Ultimately, as other universities look to bolster TA training programs, our contributions provide guidance and data points for external implementations and alterations to other training programs. Please see [1] for poster and other materials.

REFERENCES

[1] 2024. SIGCSE Poster & Materials. https://educationalbytes.com/sigcse2024

[2] Abdulmohsen H Al-Elq. 2010. Simulation-based medical teaching and learning. *Journal of family and Community Medicine* 17, 1 (2010), 35.

[3] Sue Ellen DeChenne, Natalie Koziol, Mark Needham, and Larry Enochs. 2015. Modeling sources of teaching self-efficacy for science, technology, engineering, and mathematics graduate teaching assistants. *CBE—Life Sciences Education* 14, 3 (2015), ar32.

[4] Francisco J. Estrada and Anya Tafliovich. 2017. Bridging the Gap Between Desired and Actual Qualifications of Teaching Assistants: An Experience Report. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (Bologna, Italy) *(ITiCSE '17)*. Association for Computing Machinery, New York, NY, USA, 134–139. https://doi.org/10.1145/3059009.3059023

[5] G. L. Herman, K. Trenshaw, and L. Rosu. 2012. Work in progress: Empowering teaching assistants to become agents of education reform. In *2012 IEEE Frontiers in Education Conference (FIE)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–2. https://doi.org/10.1109/FIE.2012.6462322

[6] David G. Kay. 1995. Training Computer Science Teaching Assistants: A Seminar for New TAs. *SIGCSE Bull.* 27, 1 (mar 1995), 53–55. https://doi.org/10.1145/199691.199719

[7] Elizabeth Patitsas. 2012. A Case Study of Environmental Factors Influencing Teaching Assistant Job Satisfaction. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research* (Auckland, New Zealand) *(ICER '12)*. Association for Computing Machinery, New York, NY, USA, 11–16. https://doi.org/10.1145/2361276.2361280

[8] Elizabeth Patitsas. 2013. A Case Study of the Development of CS Teaching Assistants and Their Experiences with Team Teaching. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '13)*. Association for Computing Machinery, New York, NY, USA, 115–124. https://doi.org/10.1145/2526968.2526981

[9] Ben Stephenson, Andrew Kuipers, Rosa Karimi Adl, and Flora Stephenson. 2014. Teaching Assistant in Residence: A Novel Peer Mentorship Program for Less Experienced Teaching Assistants. *J. Comput. Sci. Coll.* 29, 4 (apr 2014), 183–190.

Leveraging Kotter's 8 Stage Model of Organizational Change to Understand Broadening Participation in Computing

Kari L. George
Department of Computer Science
University of Illinois
Urbana-Champaign
Urbana, IL, USA
klgeorge@illinois.edu

Max Fowler
Department of Computer Science
University of Illinois
Urbana-Champaign
Urbana, IL, USA
mfowler5@illinois.edu

Vidushi Ojha
Department of Computer Science
University of Illinois
Urbana-Champaign
Urbana, IL, USA
vojha3@illinois.edu

Morgan M. Fong
Department of Computer Science
University of Illinois
Urbana-Champaign
Urbana, IL, USA
mmfong2@illinois.edu

Kathleen Isenegger
Department of Computer Science
University of Illinois
Urbana-Champaign
Urbana, IL, USA
kti3@illinois.edu

Christopher Perdriau
Department of Computer Science
University of Illinois
Urbana-Champaign
Urbana, IL, USA
chp5@illinois.edu

Mariam Saffar Perez
Department of Computer Science
University of Illinois
Urbana-Champaign
Urbana, IL, USA
mariams5@illinois.edu

Yael Gertner
Department of Computer Science
University of Illinois
Urbana-Champaign
Urbana, IL, USA
ygertner@illinois.edu

Colleen M. Lewis
Department of Computer Science
University of Illinois
Urbana-Champaign
Urbana, IL, USA
ColleenL@illinois.edu

ABSTRACT

Broadening participation in computing (BPC) is a focus in industry and academia. Extant research focuses on *what* broadening participation in computing (BPC) efforts are pursued, while we propose focusing on *how* change happens. Our qualitative study applied John Kotter's (2012) eight-stage change framework to analyze interviews with faculty and staff engaged in BPC efforts. Illustrative examples from our interviews elucidate each of the eight stages and how they can be applied to pursue organizational change efforts that support BPC.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; **Computer science education**.

KEYWORDS

broadening participation in computing; organizational change; diversity, equity, and inclusion

ACM Reference Format:
Kari L. George, Max Fowler, Vidushi Ojha, Morgan M. Fong, Kathleen Isenegger, Christopher Perdriau, Mariam Saffar Perez, Yael Gertner, and Colleen M. Lewis. 2024. Leveraging Kotter's 8 Stage Model of Organizational Change to Understand Broadening Participation in Computing. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635541

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635541

1 BACKGROUND

Motivated by the need for inclusion to address both the social and economic opportunities of a diverse technological workforce [1], BPC efforts aim to increase the representation of people who identify as women, African American or Black, Hispanic or Latinx/a/o/e, Native American, Indigenous, and persons with disabilities [1]. Research on BPC efforts has illuminated a variety of approaches and initiatives that seek to support BPC goals such as curricular changes in introductory computing courses [5] and exemplar institutions [2, 3]. While prior work exemplifies *what* types of work computing departments have undertaken and the outcomes of these initiatives, people from underrepresented groups remain severely underrepresented in computing [7]. Hence, we propose refocusing on *how* change happens to increase our collective capacity for impactful change efforts. Applying a change framework allows us to examine how change occurs in computer science departments, who contributes to this work, and what motivates the work.

2 KOTTER'S CHANGE FRAMEWORK

We apply John Kotter's [4] eight-stage framework of leading change, to understand how BPC change occurs, who contributes, and what motivates the changes. This framework highlights the need to challenge or dismantle the existing status quo and to introduce and embed new practices into the culture [4].

3 METHODS

Using a practitioner research approach [6], we conducted interviews with 13 faculty and staff members across R1 and one R2 U.S. institutions via Zoom. Participants were recruited based on their BPC involvement. Audio files were transcribed and anonymized. We used inductive and deductive analytic coding.

4 OVERVIEW OF RESULTS

4.1 Stage 1: Establishing a sense of urgency

Stage 1 identifies two complementary goals of *removing complacency* and *creating urgency* to motivate the work of creating change. By ensuring that there are efforts to fight complacency and create a sense of urgency, a department can initiate change efforts. Kotter warns that skipping this or other stages may threaten the success of change efforts. Participants described how both internal forces (e.g., students' demands) and external forces (e.g., the murder of George Floyd) created urgency to engage in BPC and reduced complacency.

4.2 Stage 2: Creating a guiding coalition

Kotter emphasizes the need for a guiding coalition to lead change efforts and argues they need expertise, positional power, external trust, and internal buy-in and trust to effectively lead. Our participants leveraged coalitions of department chairs, faculty, staff, and students to use their connections and social capital to enact change. Within the coalition, participants spoke about developing internal buy-in and trust, such as letting student members drive the agenda.

4.3 Stage 3: Developing a vision and strategy

Kotter highlights the importance of crafting a guiding change vision to target and scope a change process that must be communicable, imaginable, flexible, desirable, focused, and feasible. Some participants include details on how they align their goals to reach their state's demographics or craft targeted sets of recommendations from existing strategic plans. Our participants' experiences demonstrate the collaborative, messy, iterative process of vision creation.

4.4 Stage 4: Communicating the change vision

Kotter argues for giving the broader organization time to process, adapt, and adjust the change vision. By involving stakeholders in abundant communication, Kotter argues the vision will be more easily understood, taken up, and bought in by the organization more broadly. Participants exemplified this stage by soliciting community feedback on their change vision and allowing ample time for individuals to reflect on the change vision.

4.5 Stage 5: Empowering broad-based action

Once the change vision is solidified, Kotter emphasizes the importance of empowering employees (i.e., faculty, students, and staff) to support change efforts through honest dialogue, training, and aligning incentives. Examples from one institution show that repeated discussions and scheduled training in diversity led to faculty feeling invested in change efforts. In another institution, funding incentives motivated a faculty member to get involved with BPC who had previously been unaware they could assist with the efforts.

4.6 Stage 6: Generating short-term wins

In this stage, Kotter emphasizes the need to plan for short-term wins and make deliberate progress towards them to highlight that progress is being made. Publicizing efforts ensures that supporters and skeptics are aware of the success of the endeavor. Our participants described examples of short-term wins, such as creating BPC-focused academic programs and fellowships.

4.7 Stage 7: Consolidating gains

Kotter highlights the importance of maintaining and building momentum by consolidating gains made in previous stages to pursue more change. Examples from one institution include both student and faculty-driven change efforts that built capacity to pursue more change. Kotter emphasizes that building momentum is critical in the later change stages as resistance and exhaustion can threaten to undermine progress and may lead to a regression if efforts are eased before changes are fully realized.

4.8 Stage 8: Anchoring approaches in the culture

Kotter's eighth stage describes ways in which institutions can anchor new approaches into their culture; one of which is bridging old values with new values. One participant discussed the values some faculty members had and how to align those values with the department's BPC values. Understanding the *"deep set beliefs"* people have is helpful for creating cultural change because change agents may find connections between old and new values.

5 RELEVANCE TO SIGCSE COMMUNITY

We present concrete examples of departmental change from our participants that will be useful to practitioners grappling with change processes at their own institutions. The poster format allows our team to have rich conversations with SIGCSE attendees to help them understand how the framework might apply within their context.

ACKNOWLEDGMENTS

This work was partially funded by the National Science Foundation (Grants: 2030859, 2137647,1821136, 2113955, 2144249, 2229612).

REFERENCES

[1] National Science Foundation. 2012. CISE Strategic Plan for Broadening Participation. https://www.nsf.gov/cise/oad/cise_bp.jsp.
[2] Carol Frieze and Lenore Blum. 2002. Building an effective computer science student organization: the Carnegie Mellon women@SCS action plan. *ACM SIGCSE Bulletin* 34, 2 (June 2002), 74–78. https://doi.org/10.1145/543812.543835
[3] Maria Klawe. 2013. Increasing Female Participation in Computing: The Harvey Mudd College Story. *Computer* 46, 3 (March 2013), 56–58. https://doi.org/10.1109/MC.2013.4
[4] John P Kotter. 2012. *Leading Change.* Harvard Business Review Press.
[5] Sukanya Kannan Moudgalya, Chris Mayfield, Aman Yadav, Helen H. Hu, and Clif Kussmaul. 2021. Measuring Students' Sense of Belonging in Introductory CS Courses. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education.* ACM, Virtual Event USA.
[6] Sharon M Ravitch and Nicole Mittenfelner Carl. 2016. Qualitative research: Bridging the conceptual, theoretical, and methodological.
[7] Stuart Zweben and Betsy Bizot. 2023. 2022 CRA Taulbee Survey: Record doctoral degree production; more increases in undergrad enrollment despite increased degree production. https://cra.org/crn/wp-content/uploads/sites/7/2023/05/2022-Taulbee-Survey-Final.pdf.

Belonging in POGIL Gateway Classes

Olga Glebova
olga.glebova@uconn.edu
University of Connecticut
Storrs, CT, USA

Charity Lovitt
lovittc@uw.edu
University of Washington Bothell
Bothell, WA, USA

Giselle Mak
hmak@uw.edu
University of Washington Bothell
Bothell, WA, USA

Darian Castillo
cast09@uw.edu
University of Washington Bothell
Bothell, WA, USA

Matthew Horn
hornma@uvu.edu
Utah Valley University
Salt Lake City, UT, USA

Tracey Murray
tmurray2@capital.edu
Capital University
Columbus, OH, USA

ABSTRACT

The sense of belonging among 285 students was explored at four post-secondary institutions in the United States in introductory courses in Computer Science (one course) and Chemistry (three courses). In all four courses instructors employed Process Oriented Guided Inquiry Learning (POGIL), a collaborative and student-centered teaching approach. To assess students' sense of belonging, they were asked to complete surveys of seven Likert-scale questions and two open-response questions at the beginning and the end of the Spring'23 semester.

The results were analyzed quantitatively and qualitatively. For the Likert questions, no statistically significant differences were found between pre and post scores. However, there was a statistically significant correlation between a sense of belonging at the end of the semester and enrollment in the next course of the sequence. The cohort as a whole indicated that social interaction(s) is the main factor that contributed to their sense of belonging, while course characteristics and perceived self-competence contributed most to feelings of unbelonging. In addition, in the CS course, the sentiment about career goals appeared more frequently than in Chemistry classes.

KEYWORDS

Process Oriented Guided Inquiry Learning, POGIL, CS1, STEM, Sense of Belonging, Sense of Unbelonging, Gateway Class

ACM Reference Format:
Olga Glebova, Charity Lovitt, Giselle Mak, Darian Castillo, Matthew Horn, and Tracey Murray. 2024. Belonging in POGIL Gateway Classes. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V.2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635585

1 INTRODUCTION

Only 40 percent of students who enter STEM majors end up earning a STEM degree [7].

An increased sense of belonging in STEM can increase persistence in STEM classes and majors [1–3].

Thus, it is important to develop classroom techniques that increase students' sense of belonging in STEM classes, especially in early "gateway" classes. Gateway classes are taken by all students in a major early in their college careers and are often seen as difficult or as barriers to success in a major. Examples of gateway classes are introductory computer science and general chemistry.

POGIL is a teaching philosophy that uses collaborative learning and guided inquiry activities to allow students to construct their own knowledge. Because students are working collaboratively, it could be presupposed that students would build a sense of community with their classmates. This could lead to an increased sense of belonging in the class and subject. However, students who have social anxiety or identify with underrepresented populations might find the required interactions of a POGIL classroom stressful [6] which might lead to a decreased sense of belonging.

To understand the dynamic in our classrooms, we considered the following research questions. RQ1: To what degree does taking a gateway science class using POGIL methodology affect the development of science belonging? RQ2: In POGIL classes, to what extent does the development of science belonging correlate with persistence?

The theoretical framework that guided this study was self- determination theory, which presupposes that "when students' fundamental needs are met, they engage more energetically and learn more which contributes to a positive identity in science" [8]. In this framework, the POGIL classroom setting would affect the "Supportive Relationships Instructors & Peers" aspect of a student's development of identity as a scientist and belonging in the community of scientists working in their major.

2 METHODS

This study focused on the sense of belonging among a cohort of students from four U.S. post-secondary institutions, comprising 139 students enrolled in an introductory Computer Science course and 223 students in introductory Chemistry courses. Quantitative and qualitative data were collected using pre- and post-surveys. We adapted course-level belonging measures from [4] and added our own free response questions (Q8, Q9).

Q1 I feel like I fit in my course
Q2 I feel comfortable with my peers and classmates in my course
Q3 I feel comfortable with my professor in my course

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635585

Q4 I feel comfortable with my peer-facilitators/TAs in my course
Q5 Setting aside my performance in class, I feel like I belong in my course
Q6 I feel uncertain about my belonging in my course (i.e. sometimes I feel that I belong, sometimes I don't)
Q7 When I don't perform well, I feel like maybe I don't belong in my course
Q8 Describe at least one example of something that happened that made you feel like you belong in the course
Q9 Describe at least one example of something that happened that made you feel like you do NOT belong in the course

The free-response questions were analyzed using a coding framework we developed for this study, which was based on [5]. This analysis was conducted by co-authors who did not serve as instructors for any of the courses involved in the study. The coding framework encompassed eight primary categories: Course Characteristics, Individual Interest, Perceptions, Perceived Ability, Value, Student characteristics, Nothing/NA, and Social interaction.

3 RESULTS

Quantitative findings are showcased for the Chemistry courses, while qualitative outcomes are detailed for the Computer Science course. This is due to the kind of data we were able to receive and analyze by date.

Statistical analysis using a two-tailed T-test revealed no statistically significant differences in the scores of quantitative questions Q1-Q7 between the beginning and end of the semester for the entire Chemistry cohort. However, there was a statistically significant correlation between a sense of belonging at the end of the semester and enrollment in the next course of the sequence in all Chemistry courses.

For the CS course, we analyzed the qualitative responses to questions Q8 and Q9. As presented in Figure 1, the top contributor to the sense of belonging was Social Interaction, whereas Perceived Ability was the major contributor to the sense of unbelonging. Notably, a lot of students could not think of an example of what made them feel like they did not belong in a course.

Two representative quotes from CS students about what made them feel like they belong in a class are: "*When I participated in a class discussion, I felt understood and supported by my classmates and instructor*"; and "*One example of something that happened was my connection towards the subject itself, I really love coding and I would like to base my career with a great foundation from it*".

Table 1 provides a detailed breakdown of the subcategories of Social Interactions among Computer Science students. The most significant factors in developing a sense of belonging are interactions with teammates and classmates. This validates our original assumption that a class where a collaborative teaching technique is used should positively contribute to the perceived sense of belonging.

In conclusion, social interaction(s) are perceived by students to be major positive contributors to their sense of belonging in introductory gateway courses.

4 ACKNOWLEDGMENTS

This project was supported by the NSF grant 1954427, we thank NSF for their generous funding. We would also like to thank The POGIL Project and POGIL DBER project leaders for their valuable consultations.

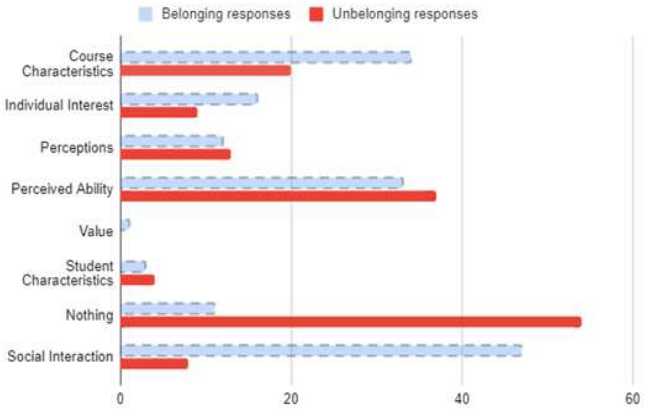


Figure 1: CS specific contributors to senses of belonging and unbelonging

Table 1: Summary of various social interactions

	# belonging responses	# unbelonging responses
Total Social interaction	**47**	**8**
Teaching assistant(s)	1	0
Professor	7	1
Classmates	17	1
Team	22	6
External figures	0	0

REFERENCES

[1] Thomas P Clements, Katherine L Friedman, Heather J Johnson, Cole J Meier, Jessica Watkins, Amanda J Brockman, and Cynthia J Brame. 2022. "It made me feel like a bigger part of the STEM community": Incorporation of Learning Assistants Enhances Students' Sense of Belonging in a Large Introductory Biology Course. *CBE life sciences education* 21, 2 (June 2022), ar26. https://doi.org/10.1187/cbe.21-09-0287

[2] Anne Deiglmayr, Elsbeth Stern, and Renate Schubert. 2019. Beliefs in "Brilliance" and Belonging Uncertainty in Male and Female STEM Students. *Frontiers in Psychology* 10 (2019). https://doi.org/10.3389/fpsyg.2019.01114

[3] Tara C Dennehy and Nilanjana Dasgupta. 2017. Female Peer Mentors Early in College Increase Women's Positive Academic Experiences and Retention in Engineering. *Proceedings of the National Academy of Science* (2017).

[4] Angela Fink, Regina F. Frey, and Erin D. Solomon. 2020. Belonging in general chemistry predicts first-year undergraduates' performance and attrition. *Chem. Educ. Res. Pract.* 21 (2020), 1042–1062. Issue 4. https://doi.org/10.1039/D0RP00053A

[5] Angela Fink, Jessica D. Young, Neil K. Vuppala, and Regina F. Frey. 2023. Mixed-methods exploration of students' written belonging explanations from general chemistry at a selective institution. *Chem. Educ. Res. Pract.* 24 (2023), 327–352. Issue 1. https://doi.org/DOIhttps://doi.org/10.1039/D2RP00166G

[6] Virginia R Downing Katelyn M Cooper and Sara E Brownell. 2018. The influence of active learning practices on student anxiety in large-enrollment college science classrooms. *International Journal of STEM Education* (2018).

[7] President's Council of Advisors on Science and Technology. 2012. "Engage to Excel: Producing One Million Additional College Graduates with Degrees in Science, Technology, Engineering, and Mathematics. (February 2012). https://obamawhitehouse.archives.gov/sites/default/files/microsites/ostp/pcast-engage-to-excel-final_2-25-12.pdf

[8] Ellen Skinner, Emily Saxton, Cailin Currie, and Gwen Shusterman. 2017. A motivational account of the undergraduate experience in science: Brief measures of students' self-system appraisals, engagement in coursework, and identity as a scientist. *International Journal of Science Education* (2017).

CAET: Code Analysis and Education Tutor

Colin Glynn
Computer Science Department
College of Saint Benedict and Saint John's University
Saint Joseph, Minnesota, USA
cglynn001@csbsju.edu

Emily Hed
Computer Science Department
College of Saint Benedict and Saint John's University
Saint Joseph, Minnesota, USA
ehed001@csbsju.edu

Abbigail Pexa
Computer Science Department
College of Saint Benedict and Saint John's University
Saint Joseph, Minnesota, USA
apexa001@csbsju.edu

Tyler Pohlmann
Computer Science Department
College of Saint Benedict and Saint John's University
Saint Joseph, Minnesota, USA
tpohlmann001@csbsju.edu

Imad Rahal
Computer Science Department
College of Saint Benedict and Saint John's University
Saint Joseph, Minnesota, USA
irahal@csbsju.edu

Robert Hesse
Computer Science Department
College of Saint Benedict and Saint John's University
Saint Joseph, Minnesota, USA
rhesse@csbsju.edu

ABSTRACT

The introduction of OpenAI's ChatGPT in 2022 kickstarted the release of Generative Artificial Intelligence (GAI) applications to the public domain. Such chat interfaces are based on large language models (LLMs) and possess a vast array of abilities spanning conversation, the writing and debugging of code, the writing of papers, and the creation of images, music, and songs. With students now having access to a myriad of GAI tools, academia has been permanently altered.

Our proposed system, named Code Analysis and Education Tutor (CAET), integrates GAI into early Computer Science education by providing students with an ethical alternative to existing GAI tools. CAET is designed to assist students with programming tasks in a manner tailored to their individual needs without jeopardizing the integrity of their learning. A point of uniqueness from existing works is CAET's ability to display or hide generated code based on its pertinence to the problem at hand. After subjecting multiple GAI models to common programming errors and queries, we settled on OpenAI's GPT-3.5 Turbo model due to its comprehensive capabilities and cost-effectiveness. Overall, CAET underscored the model's conversational dynamics and provided insights for creating a more personalized learning experience for students in an introductory computer science course.

ACM Reference Format:

Colin Glynn, Emily Hed, Abbigail Pexa, Tyler Pohlmann, Imad Rahal, and Robert Hesse. 2024. CAET: Code Analysis and Education Tutor. *In SIGCSE '24: ACM Technical Symposium on Computer Science Education, March 20–23, 2024, Portland, OR.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635543

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA © 2024 Copyright is held by the owner/author(s). ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635543

1 PROPOSED SYSTEM

Impacts of GAI tools have had far-reaching consequences in both academia and the software industry. A recent system called CodeHelp [3] shows the ability to artificially generate responses to student queries by manually filtering out code and returning code explanations in plain text. In industry, GitHub CoPilot [2] has been gaining in popularity among software developers to help auto-complete sections of code for developers. Obviously, these two systems lie on opposite ends of the "programming aid" spectrum; a major contribution of our system is the ability to combine features of both.

In response to a user query, CAET modifies responses received from ChatGPT [1] using a separate context layer that tunes the model to specific requirements and expectations. This layer is aware of the programming problems assigned and ensures responses are in line with the students' backgrounds. Although the system is still undergoing testing and tuning, early results indicate that CAET can produce accurate and helpful answers and code snippets in response to basic programming queries. However, it also proved too helpful in certain instances and generated a significant portion of the solution especially when given multiple follow-up prompts. As a result, we integrated a response filtering component into the context layer which examines all code generated in comparison to a bank of solutions in order to decide when to hide or show generated code snippets.

1.1 System Overview

Rather than providing it as an external resource, CAET is implemented as an IDE extension to promote its active and convenient role in the student's development process. Figure 1 shows the overall process of using CAET along with help

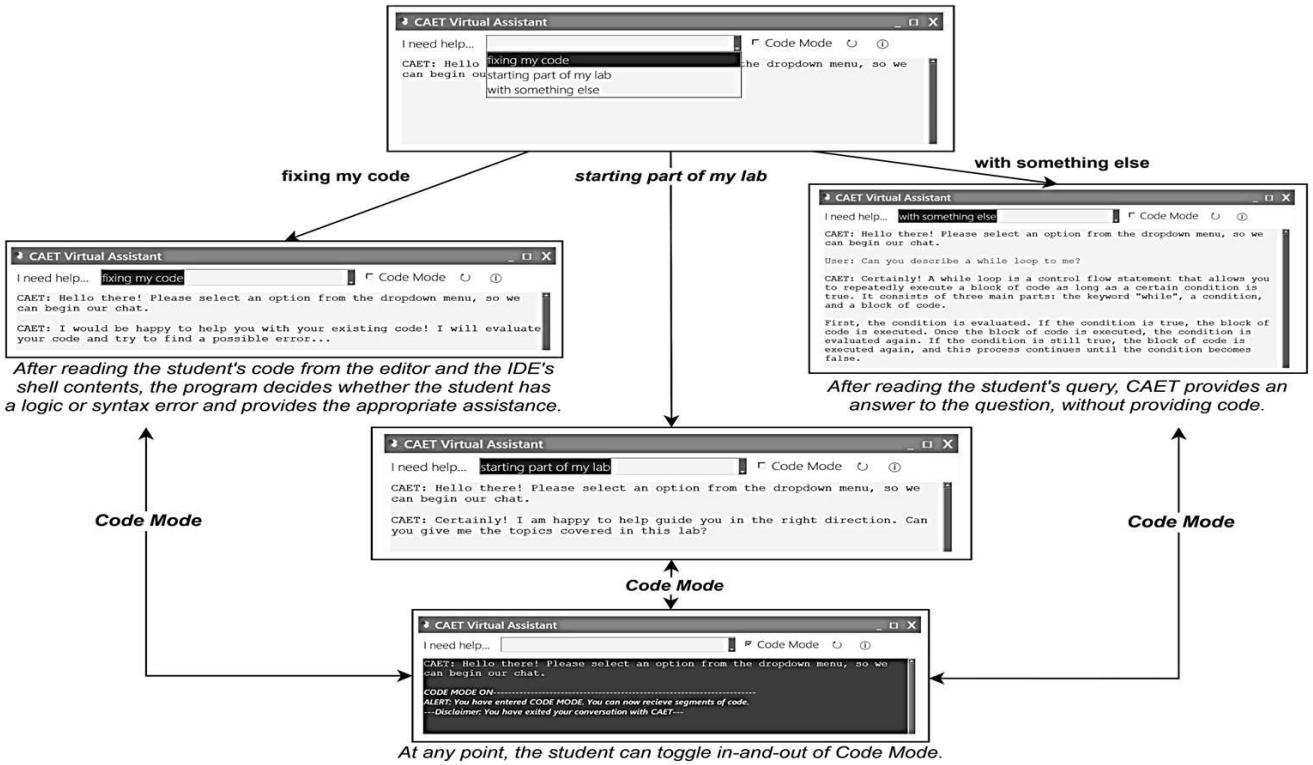


Figure 1: CAET Virtual Assistant

options available. It consists of a chat interface where the user enters a query and receives an artificially generated response. The user starts by selecting the type of assistance desired from a dropdown menu at the top of their screen. As users interact with CAET, it remembers previous queries and responses allowing for a conversation style of assistance as the user can continue to prompt with additional questions until satisfied. Categories for the types of responses being generated are limited by the specific requirements and expectations specified in the context layer. For our purposes, the context layer tunes the model toward introductory Python programming preventing it from providing advanced or direct solutions in the form of code.

1.2 Response Filtering

A critical aspect of our system is to be able to provide snippets of code to the user when working in *Code Mode* (see Figure 1). We believe this to be appropriate only if it does not directly progress the student toward the solution. Therefore, code detected in a response generated by CAET in *Code Mode* is reviewed before it is displayed to the user. In addition, *Code Mode* prohibits CAET from remembering previous queries to avoid the "multiple follow-up prompts" problem discussed earlier.

The filtering process commences by comparing the lengths of the generated code to the expected solution. If their respective lengths differ by 20% or less, a cosine similarity score [5] is calculated after converting each to a term frequency vector (Term Frequency by Inverse Document Frequency (TF-IDF) [4]). In short, TF-IDF assigns a relevance value to each term in a text; the resulting vectors are then compared using cosine similarity.

A different evaluation strategy is employed for instances where the lengths of CAET's generated code and the problem's solution differ by more than 20%. A major concern here is that the generated code may still be a segment that exists within the full solution. Comparison for this instance involves using a version of pairwise local sequence alignment [6] and assigning a score to how well the produced code fits within the solution. The threshold determined for this similarity was set to be 50%.

All thresholds discussed here were derived experimentally. In the latter case, multiple test cases were conducted to determine the 50% threshold value. For each test case, the chosen solution code was broken, given to ChatGPT to fix, and then the two codes were compared, receiving a similarity score; generated code receiving scores below this is shown to the user. At this early stage, we hope that our participation in a poster session will be helpful in collecting valuable feedback from the community prior to engaging in a detailed study.

REFERENCES

[1] ChatGPT. 2022. https://openai.com/blog/chatgpt
[2] Copilot. 2023. https://github.com/features/copilot
[3] Mark Liffiton, Brad Sheese, Jaromir Savelka and Paul Denny. 2023. Using Large Language Models with Guardrails for Scalable Support in Programming Classes. CoRR abs/2308.0692.
[4] Claude Sammut and Geoffery Web (Ed.). 2011. Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_832
[5] Science Direct. 2019. Getting to Know Your Data. Retrieved from https://www.sciencedirect .com /topics/computer-science/cosine-similarity
[6] Wikipedia. 2023. Sequence alignment. Retrieved from https://en.wikipedia.org/w/index.php?title=Sequence_alignment&oldid=1173346873

Increasing Diversity of Computing Course Participation Using A Summer Bridge Program

Elias Gonzalez
Computer Science
University of Maryland
egonzal5@umd.edu

Katheryn Atchison
University of Maryland
katea@umd.edu

ABSTRACT

For many underrepresented groups, attending a large university and immediately being able to acclimate and succeed is a tall task. Many students come to campus interested in computing but do not have the prior experience to know what path they want to take or have the necessary content knowledge to feel capable of taking on their first computing course. Additionally, without prior experience in computing, knowing how to select a specific computing major and career is difficult to do while already feeling behind.

The Break Through Tech Guild program at the University of Maryland is a summer bridge program that is a key part of a national initiative to close the gender gap in computing majors on large university campuses. This specific program runs prior to the start of the semester for one week at a time. In the past three years, the program has served over 200 students from countless different majors and backgrounds, with the explicit goal of giving them the tools and community to be able to succeed in Computing. Students learn core coding concepts from a Computer Science instructor, tackle a real-world design problem, and explore careers in tech both through mentorship from industry professionals and from the campus. We have noticed significant amounts of students have continued to computing courses after the program.

ACM Reference Format:
Elias Gonzalez and Katheryn Atchison. 2024. Increasing Diversity of Computing Course Participation Using A Summer Bridge Program. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635634

1 WHAT IS THE SUMMER BRIDGE PROGRAM?

The summer bridge program is a one-week, full-day paid experience aimed at students with little to no prior experience in computing. The goal of this program is to help build technical skills among students from populations historically excluded in the field (e.g., BLNA, Women/Non-binary, First generation), especially those with little to no prior computing experience. When in less-inclusive colleges, one in four women feel like an outsider. Just one in 20 women feel that way in more inclusive colleges, we know this is a challenge many students face [1]. Prior to the start of the semester, students spend one week together with academic content, field trips, and community events tied together around a solutions-driven theme. During the full-day program, students complete 3 core aspects: coding concepts, design thinking, and career exploration. The week is run with a core theme in mind, framing the examples and issues they are solving throughout the week.

For core coding concepts students are taught about variables, conditionals, and event-driven programming among other core concepts for app design. Students are taught these concepts in either Python or the Thunkable app development environment. Every day students complete small check-in assignments to ensure they learned the necessary material before the next day. This portion is meant to tackle students feeling prepared for their computing courses, many of which are geared toward students with much more experience–even at the introductory level– adversely affecting our underrepresented groups [2].

In the design thinking portion of the course, students are given a problem statement related to the theme at the beginning of the week that they will be looking to tackle as the week goes by. During this portion of each day, students are put in groups and are coached by industry partners through the design thinking process. Each day the students are tackling a different aspect of the process and tasked with creating a deliverable that leads to solving a real-world problem. In 2021, students were tasked with solving transportation issues on campus, in 2022, they were given the theme "Rebuilding Social Ties on Campus" and this year they were given the theme "Financial Technology." Students were expected to present their apps to both their peers and mentors as well as other key university stakeholders at the end of the week.

Lastly, for career exploration the students are exposed to all entry points into computing on campus, academic and we have six different computing majors that we work with on campus across four different colleges. In addition to talking about academic programs on campus, students have visits from the career center and have a career panel where they ask questions relating to getting into computing jobs and what options are available to them.

1.1 DEMOGRAPHICS OF ATTENDEES

The University of Maryland hosted its first summer bridge program in August 2021 with 54 students enrolled and 51 students who completed the program. Students completed the one-week program virtually with 30 students per session. They were led by two instructors and three teaching assistants. 84% of these students were women, transgender, nonbinary/gender-fluid/third gender while 21.5% identified as Black, Latina, and Native American (BLNA).

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635634

50% of these students were undeclared. In 2022 the University of Maryland hosted three sessions with 92 students enrolled and 86 students who completed the program. Two sessions were held virtually, and one session was held in person with 30 students per session. They were led by two instructors and three teaching assistants. 80% of these students were women, transgender, nonbinary/gender-fluid/third gender while 34% identified as BLNA. 50% of these students were undeclared. In 2023 University of Maryland hosted four sessions with 97 students enrolled and 85 students who completed the program. Two sessions of the program were held virtually, and two sessions were held in person with 30 students per online session and 20 students per in-person session. They were led by two instructors and 4 teaching assistants. 89 identified as an underrepresented gender (woman, non-binary, gender-fluid, third gender), and 29 identified as BLNA. The poster introduces the summer bridge program and presents positive benefits for the students who participate in it.

2 OUTCOMES

The broad goal of the program is to increase the percentage of women in our computing majors on campus. To track successes, we have taken a pre and post-survey of the students to see how many of the students are now looking to be in a computing course (in one of our partner computing programs, spanning across at least three colleges). This survey was a requirement for the students to receive their stipend for the week. In the first year of the program, 2021, we already noticed that there was extremely positive feedback and data with:

- 82% of our students who consented to having their academic outcomes tracked took at least one computing course during the '21-'22 academic year.
- 89% of students who consented to participate in research are somewhat likely, likely, or very likely to complete a major in computing.
- 100% are somewhat likely, likely, or very likely to pursue a career that requires computing knowledge and skills.

This was much higher than we were expecting. In 2022, we found that of the 73 students who responded to our survey and consented to participate in the research:

- 84% are somewhat likely, likely, or very likely to complete a major in computing.
- 84% are planning on registering for a computing course in fall 2022 or spring 2023.
- 95% are likely or very likely to pursue a career that requires computing knowledge and skills.

In 2023, of the 55 students who responded to our survey and consented to participate in research:

- 95% are somewhat likely, likely, or very likely to complete a major in computing.
- 93% are planning on registering for a computing course in fall 2023 or spring 2024.
- 98% are somewhat likely, likely, or very likely to pursue a career that requires computing knowledge and skills.

Overall, across the two years with at least one full semester of data (2021 and 2022), the number of students who have completed computing courses after completing the program exceeds 85%. This number is especially significant because 24 different majors participated in both 2021 and 2022. The students were in 9 of the 12 different colleges on campus. Beyond just taking their initial computing course, we have also seen a high rate of students continuing in Computing majors. Of our 2021 students, 78% percent of students have taken at least 3 different computing courses. In addition, of our 2022 cohort of students, 71% percent of students have taken at least 2 computing courses.

Overall, we can see that this program has led to a high rate of our students continuing into a computing major which they may not have joined without the program.

3 FUTURE DISCUSSION

As the program grows and students further matriculate into the university, we plan to track their progression and success. Specific interest in differences in both graduation rates; time to graduation, as well as career paths will be tracked. In the short term, we plan to track how impactful students continuing to take one computing course is on their long-term interest in tech. From there we will be able to track how many of the students become computing majors as well. Additionally, many students go on to join our larger efforts at our institution in DEI. This additional and larger support structure should be studied and considered when mapping out retention and graduation rates.

REFERENCES

[1] Accenture. 2020. Resetting Tech Culture. (2020). https://www.accenture.com/content/dam/accenture/final/a-com-migration/pdf/pdf-134/accenture-a4-gwc-report-final1.pdf#zoom=50

[2] Eileen D. Bunderson and Mary Elizabeth Christensen. 2014. An Analysis of Retention Problems for Female Students in University Computer Science Programs. *Journal of Research on Computing in Education* (Feb. 2014), 18 pages. https://doi.org/10.1080/08886504.1995.10782148

Engaging Students from Rural Communities in AI Education with Game-Based Learning

Alex Goslen
North Carolina State University
Raleigh, NC, USA
amgoslen@ncsu.edu

Anisha Gupta
North Carolina State University
Raleigh, NC, USA
agupta44@ncsu.edu

Smrithi Muthukrishnan
North Carolina State University
Raleigh, NC, USA
smuthuk3@ncsu.edu

Raven Midgett
North Carolina State University
Raleigh, NC, USA
rcmidget@ncsu.edu

Wookhee Min
North Carolina State University
Raleigh, NC, USA
wmin@ncsu.edu

Jessica Vandenberg
North Carolina State University
Raleigh, NC, USA
jvanden2@ncsu.edu

Veronica Cateté
North Carolina State University
Raleigh, NC, USA
vmcatete@ncsu.edu

Bradford Mott
North Carolina State University
Raleigh, NC, USA
bwmott@ncsu.edu

ABSTRACT

As the presence of artificial intelligence (AI) technologies increases throughout everyday life, so does the need to engage rural communities in AI learning experiences, as these communities often have limited access to such educational opportunities. This work presents three game-based learning activities rooted in core AI concepts: natural language processing, search, and reinforcement learning. These activities were implemented in a summer camp with middle grades students in a rural area of the USA. We share an overview of the activities, as well as key observations and takeaways from student responses in post-activity surveys.

ACM Reference Format:
Alex Goslen, Anisha Gupta, Smrithi Muthukrishnan, Raven Midgett, Wookhee Min, Jessica Vandenberg, Veronica Cateté, and Bradford Mott. 2024. Engaging Students from Rural Communities in AI Education with Game-Based Learning. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635549

1 INTRODUCTION

Recent advancements in AI in combination with the rapid adoption of these technologies have highlighted the need for K-12 AI education. Rural students are often at a disadvantage because they have limited access to these technologies as well as educational experiences that introduce them to AI concepts [1, 3].

To help address this disparity, our work presents three game-based learning activities that aim to teach concepts from core areas of AI [4]: natural language processing (NLP), search, and reinforcement learning (RL). These learning activities were incorporated into a five-day, AI-focused summer camp for middle grades students. The camp was held in a rural community of the United States, and a total of 10 students participated. Students interacted with each learning activity for approximately 30 minutes and completed a post-activity survey. We present a description of each activity and the key takeaways from students' survey responses.

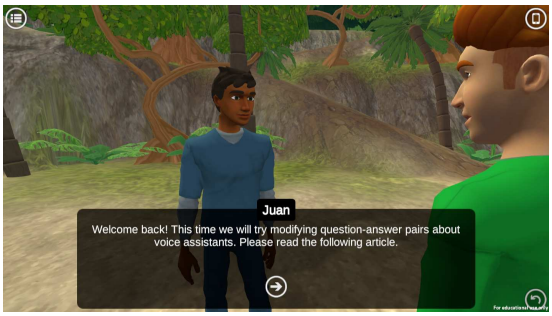


Figure 1: NLP activity.

2 NLP ACTIVITY

One day of the camp focused on exploring "Natural Interaction" with AI systems. During this day, students completed an NLP-based chatbot activity. Prior to this activity, students were introduced to several NLP applications, such as speech recognition and sentence generation, and interacted with real-world applications like Siri. Students then engaged with the NLP game-based learning activity, where they were tasked with building a non-player character (NPC)'s dialogue (Figure 1). The interactions with the NPC followed a Use-Modify-Create scaffolding progression [2]. The first phase (*use*) allowed students to view example question-answer (QA) pairs and then ask the NPC various questions. Next, students could *modify* some of the answers to provided questions to see the impact of their changes. In the final phase, students were able to *create* their own questions and answers and see how the NPC's dialogues could change based on student-authored QA pairs.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635549

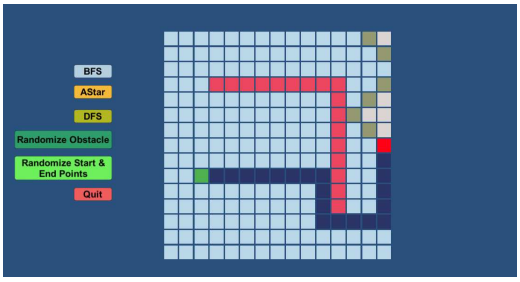


Figure 2: Search activity.

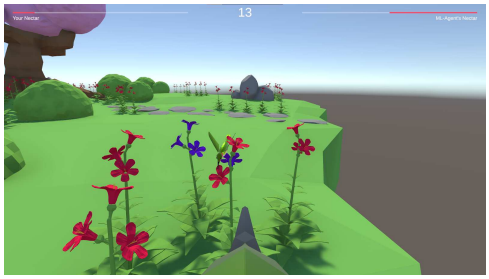


Figure 3: Reinforcement learning activity.

From the survey, students understood voice assistants' ability to process natural language. However, when asked for challenging scenarios for an AI system, students responded with examples of non-NLP tasks, such as taste food or have a physical presence. This indicates that students developed an understanding of the capabilities of AI, but not necessarily limitations of NLP. We could improve this understanding by incorporating nuances of natural language into the dialogue activity.

3 SEARCH ACTIVITY

We incorporated our search-based pathfinding activity into the summer camp day centered on "Perception" in AI systems. Students were first introduced to pathfinding using real-world examples (i.e., Google Maps). Then, we presented our game-based maze activity that incorporated three common pathfinding algorithms: Depth-First Search, Breadth-First Search, and A* Search. This activity was based on an existing open-source project[1], which we revised to meet our needs. Students were provided with a 2D grid where they could dynamically generate the start point, end point, and obstacles (Figure 2). Then, they could choose from the different algorithms to see how they explored the search space.

Based on the survey responses, real-world examples—particularly autonomous vehicles—resonated with students. Students generally answered that A* was most efficient, but were unable to provide detailed reasoning. Common responses, such as "not waste any more time," indicated that while students understood the concept of efficiency, a clear comparison of algorithms was not evident. Future improvements will include adding self-driving cars into the activity to connect to real-world examples (e.g., creating a scenario where a car needs to get from one point to another). Additionally, incorporating rewards to distinguish the applicability of algorithms in different scenarios may help further improve students' understanding of efficiency while promoting their engagement.

4 REINFORCEMENT LEARNING ACTIVITY

The reinforcement learning (RL) activity was incorporated into the "Machine Learning" day of the camp. An introductory presentation explaining RL and the concepts of agents, rewards, and exploitation and exploration was initially shown to students. They then interacted with the game-based reinforcement learning activity, built on an existing public Unity project[2], where they acted as a hummingbird trying to collect nectar from flowers (Figure 3). In the game, students contend with an adversarial agent, another hummingbird, and must gather nectar from more flowers than the opponent. There were three levels of increasing difficulty in the game, with more difficult levels having an agent trained longer using RL.

Students were very engaged with the activity, often being vocal about wanting to beat the other hummingbird. Based on the survey, students were able to understand that the time spent training the adversarial hummingbird affected its performance. However, not many students associated negative rewards with performance. Additionally, several students were unable to define the agent in this setting and thought they were training the hummingbird through gameplay. Future improvements will include drawing a distinction between training the adversarial agent by itself and playing against an opponent. Furthermore, including more visualization of key concepts like rewards and exploration vs. exploitation could help.

5 CONCLUSION

Overall, students reported enjoying the activities and showed a high level of engagement in the activities. For future studies, we aim to incorporate AI ethics concepts into the materials and examine learning gain to assess the potential impact these materials can have for rural communities.

ACKNOWLEDGMENTS

This research was supported by funding from the National Science Foundation under grant DRL-2148680. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] Andrew Crain and Karen Webber. 2021. Across the urban divide: STEM pipeline engagement among nonmetropolitan students. *Journal for STEM Education Research* 4, 2 (2021), 138–172.
[2] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. Computational thinking for youth in practice. *Acm Inroads* 2, 1 (2011), 32–37.
[3] Reenay RH Rogers and Yan Sun. 2018. *Engaging STEM Students From Rural Areas: Emerging Research and Opportunities: Emerging Research and Opportunities.* IGI Global, Hershey, PA.
[4] David Touretzky, Christina Gardner-McCune, Bryan Cox, Judith Uchidiuno, Janet Kolodner, and Patriel Stapleton. 2022. Lessons Learned From Teaching Artificial Intelligence to Middle School Students. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2.* 1371–1371.

[1] https://github.com/dbrizov/Unity-PathFindingAlgorithms

[2] https://learn.unity.com/course/ml-agents-hummingbirds

Supporting Instructors Adoption of Peer Instruction

Xingjian Gu
xjgu@umich.edu
University of Michigan
Ann Arbor, MI, USA

Barbara J. Ericson
barbarer@umich.edu
University of Michigan
Ann Arbor, MI, USA

Zihan Wu
ziwu@umich.edu
University of Michigan
Ann Arbor, MI, USA

ABSTRACT

Peer Instruction (PI) is a lecture-based active learning approach that has students solve a difficult multiple-choice question individually, submit their answer, discuss their answer with peers, and then submit their answer again. Despite plentiful evidence to support its effectiveness, PI has not been widely adopted by undergraduate computing instructors due to low awareness of PI, the effort needed to create PI questions, the limited instructional time needed for PI activities during lectures, and potential adverse reactions from students.

We hypothesized that we could allay some of these concerns by hosting a three-day summer workshop on Peer Instruction for instructors and building and sharing a free tool and a question bank that supports PI in an open-source ebook platform. We invited eighteen instructors to attend an in-person three-day workshop on PI in the summer of 2022. We collected their feedback by using pre and post surveys and conducting semi-structured interviews. We report on the effect of the three-day summer workshop on instructor attitudes towards and knowledge of PI, the barriers that prevented instructors from adopting the free tool, and feedback from instructors who used the tool.

The results show that most workshop attendees reported that they planned to use the tool in the fall semester, but less than half actually did. Responses from both users and non-users yield insights about the support instructors need to adopt new tools. This research informs future professional development workshops, tool development, and how to better support instructors interested in adopting Peer Instruction.

KEYWORDS

Peer Instruction, online textbooks, interactive textbooks, professional development

ACM Reference Format:
Xingjian Gu, Barbara J. Ericson, and Zihan Wu. 2024. Supporting Instructors Adoption of Peer Instruction. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635530

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635530

1 INTRODUCTION

In Peer Instruction an instructor displays a hard multiple-choice question in lecture. Students answer individually, typically discuss their answers with nearby peers, and answer again [3]. The instructor then leads a discussion of the question.

Decades of research has shown that Peer Instruction improves learning versus passive lecture in many fields including physics [3, 4], biology [8], and computer science [11]. In computer science it has substantially reduced failure rates [10], improved retention [12], and increased final exam performance [14]. This is true for both introductory computing courses [1] as well as advanced computing courses [9].

Prior research identifies several barriers to the successful adoption of Peer Instruction. A survey of computing instructors [1] on the SIGCSE member list with 125 respondents found that about 17% of respondents were not aware of Peer Instruction. It also found that 42% were aware of it, but had not tried it. One reason for not trying Peer Instruction was satisfaction with the current teaching approach. The second reason was the time it takes to learn about and successfully adopt Peer Instruction. Faculty typically work over 50 hours a week [7], so finding time to learn and implement a new teaching approach is a major obstacle. Most instructors who try Peer Instruction create their own questions, which is a huge part of the time needed to successfully adopt it [1].

We hypothesized that we could successfully address some of these issues by providing a three-day instructor professional development workshop in the summer of 2022. The workshop provided hands-on use of a free tool that supports Peer Instruction, which we developed and integrated into an open-source ebook platform, Runestone [6]. We added over 600 existing Peer Instruction questions for CS1 in Python, Java, and C++ that were embedded in slides at http://www.peerinstruction4cs.org/. We also added 127 questions from PeerWise [5] and 622 questions for CS1 and CS2 from the Canterbury Question Bank [13]

This poster presents our findings on the Peer Instruction workshop's effect on instructors. We explored for non-users what barriers prevented them from adopting the free tool, and for instructors who used it in at least one course after the workshop, what worked well and what could be improved.

2 METHODS

Eighteen instructors attended the workshop. During the workshop, we demonstrated how to create a new PI assignment, search for existing PI questions and add them to an assignment, author new PI questions, and had volunteer instructors practice leading a Peer Instruction session with the new tool. We also reviewed how to write a good PI question and best practices for PI.

To collect the instructors' reflections during the workshop, We administered a pre-survey on the first day of the three-day summer

instructor workshop in summer 2022 and a post-survey on the last day. Two instructors left early and did not fill out the post-survey. We administered a second post-survey after the fall semester of 2022. The results of the second post-survey were used to select eight instructors for semi-structured interviews: four who had used the tool and four who had not. We conducted a thematic analysis of the transcripts from these semi-structured interviews [2].

3 RESULTS AND FUTURE WORK

From the pre-survey most (77.8%) of the instructors had not used Peer Instruction in their classes before the workshop. On the post-survey all but one (15/16 94%) of the instructors reported that they were very likely or likely to use the tool in the fall but less than a third (5/16 31.3%) actually did. The number one reason for not trying the tool in the fall was a lack of time (8/11 72.7%). This was due to a variety of reasons including teaching a new course, new to teaching, getting COVID, family issues, a difficult situation with a new lecturer, a lack of PI questions for their particular course (discrete math), and the time required to integrate the new tool with the course. Importantly, all of the instructors who used the free tool plan to use it again.

We found that among instructors who did not use the tool, those already experienced in PI ended up using their previous method for PI after the workshop. One instructor attempted to set up the tool, but failed to get it ready in time. This highlights the need for more follow-up support. Three of the instructors chose other methods because they did not perceive the tool as easy to use compared to their existing methods, even though some methods require a paid subscription from students. To encourage the instructors to adopt a free Peer Instruction tool, as opposed to other methods, it is important to improve the tool's perceived ease of use and provide more support for instructors after the workshop,

Although the four instructors who used the tool teach in different contexts, including one in high school and three at universities, all reported that students actively participated in the discussion while using it. They also received positive feedback on the tool from their students despite some initial technical problems. In their classes, it has generally been used for engaging students and encouraging communication between students. For those reasons, the tool was used for extra credit activities, a break between lectures, or a "mini collaboration" activity between larger group activities or projects.

Meanwhile, some things could be improved. All instructors mentioned that technical problems occurred, which the team has fixed since receiving the feedback. Instructors want an easier way to author new PI questions and grade PI sessions. Instructors also want more post-workshop support and better documentation.

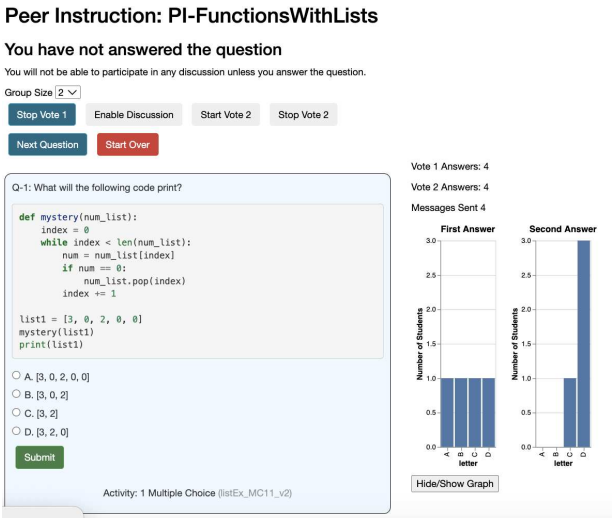


Figure 1: The Instructor Interface

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2043207. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We also thank Shefali Patel for conducting interviews and Prajakta Bonde for helping with developing the codebook and coding the transcripts.

REFERENCES

[1] Dennis Bouvier, Ellie Lovellette, John Matta, Jing Bai, Jacqueline Chetty, Stan Kurkovsky, and Jia Wan. 2019. Factors Affecting the Adoption of Peer Instruction in Computing Courses. In *Proceedings of the Working Group Reports on Global Computing Education*. 1–25.

[2] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[3] Catherine H Crouch and Eric Mazur. 2001. Peer instruction: Ten years of experience and results. *American journal of physics* 69, 9 (2001), 970–977.

[4] Catherine H Crouch, Jessica Watkins, Adam P Fagen, and Eric Mazur. 2007. Peer instruction: Engaging students one-on-one, all at once. *Research-based reform of university physics* 1, 1 (2007), 40–95.

[5] Paul Denny, John Hamer, Andrew Luxton-Reilly, and Helen Purchase. 2008. PeerWise: students sharing their multiple choice questions. In *Proceedings of the fourth international workshop on computing education research*. 51–58.

[6] Barbara J Ericson and Bradley N Miller. 2020. Runestone: A Platform for Free, Online, and Interactive Ebooks. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 1012–1018.

[7] James S Fairweather. 2005. Beyond the rhetoric: Trends in the relative value of teaching and research in faculty salaries. *The journal of higher education* 76, 4 (2005), 401–422.

[8] Jennifer K Knight and William B Wood. 2005. Teaching more by lecturing less. *Cell biology education* 4, 4 (2005), 298–310.

[9] Cynthia Bailey Lee, Saturnino Garcia, and Leo Porter. 2013. Can peer instruction be effective in upper-division computer science courses? *ACM Transactions on Computing Education (TOCE)* 13, 3 (2013), 1–22.

[10] Leo Porter, Cynthia Bailey Lee, and Beth Simon. 2013. Halving fail rates using peer instruction: a study of four computer science courses. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 177–182.

[11] Leo Porter, Cynthia Bailey Lee, Beth Simon, Quintin Cutts, and Daniel Zingaro. 2011. Experience report: a multi-classroom report on the value of peer instruction. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. 138–142.

[12] Leo Porter and Beth Simon. 2013. Retaining nearly one-third more majors with a trio of instructional best practices in CS1. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 165–170.

[13] Kate Sanders, Marzieh Ahmadzadeh, Tony Clear, Stephen H. Edwards, Mikey Goldweber, Chris Johnson, Raymond Lister, Robert McCartney, Elizabeth Patitsas, and Jaime Spacco. 2013. The Canterbury QuestionBank: Building a Repository of Multiple-Choice CS1 and CS2 Questions. In *Proceedings of the ITiCSE Working Group Reports Conference on Innovation and Technology in Computer Science Education-Working Group Reports* (Canterbury, England, United Kingdom) *(ITiCSE -WGR '13)*. Association for Computing Machinery, New York, NY, USA, 33–52. https://doi.org/10.1145/2543882.2543885

[14] Beth Simon, Julian Parris, and Jaime Spacco. 2013. How we teach impacts student learning: peer instruction vs. lecture in CS0. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 41–46.

A Study of Undergraduate Learning Assistants (ULAs) in Computer Science

Shirin Haji Amin Shirazi
University of California, Riverside
California, USA
shaji007@ucr.edu

Mariam Salloum
University of California, Riverside
California, USA
mariam.salloum@email.ucr.edu

Neftali Watkinson
University of California, Riverside
California, USA
neftaliw@ucr.edu

ABSTRACT

With the recent surge in interest in Computer Science, many challenges have emerged, ranging from faculty turnover to providing adequate resources and an inclusive environment for students throughout their academic journey. As part of our ongoing efforts, we introduce the concept of "Undergraduate Learning Assistants" (ULAs) – undergraduates carefully selected and trained by our department to serve as peer tutors for core Computer Science courses, such as CS1, CS2, and CS3. This program, driven not only by technical considerations but also by the goal of fostering a more inclusive and accessible learning community, seeks to enhance students' sense of belonging and encourage undergraduates to explore potential teaching roles. In this article, we share our experiences and present our assessments of this program.

ACM Reference Format:
Shirin Haji Amin Shirazi, Mariam Salloum, and Neftali Watkinson. 2024. A Study of Undergraduate Learning Assistants (ULAs) in Computer Science. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635630

1 INTRODUCTION

Computer science has been one of the fastest-growing fields in academia[1]. This growth has led to challenges that institutions must address to support the students effectively, such as assembling an adequate teaching workforce (instructors, teaching assistants, graders, etc.), faculty turnover, and establishing an inclusive and welcoming environment[2].

Furthermore, as the interest in the field increases, the significance of core Computer Science courses (such as CS1, CS2, and CS3) becomes more apparent in students' overall success. These courses usually have high enrollment and involve students with different academic backgrounds. Hence, teaching assistants are essential in alleviating the faculty workload and the progress and satisfaction of the participating students.

In this article, we share our experience with a peer tutoring program that employs "Undergraduate Learning Assistants" or ULAs to help students in fundamental CS courses as independent tutors. A ULA offers tutoring for specific departmental courses without any involvement in student records or grading responsibilities. While previous initiatives have reported hiring Undergraduate TAs[4] or in-lab undergrad tutors[7], to our knowledge, our ULA program is the first to extend tutoring beyond the traditional confines of course schedules. In addition, we discuss our ongoing study of the program's impact on our student community, ULAs, and faculty members. We also report our assessment methodology and results in the following sections.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635630

2 PROGRAM DESIGN

The ULA program in our institution was designed back in the Fall of 2021 with the following main goals :

- Improving the learning outcomes for students in core CS courses,
- Building a sense of community among undergraduates and,
- Creating a supportive environment for underrepresented students.

Upon reviewing applications, academic records, GPAs, and evaluating each candidate's qualifications, undergraduate students are eligible to be selected for ULA positions in the department where they will serve as tutors and provide help with CS courses (including but not limited to Introduction to Computer Science I&II, Introduction to Data Science and Algorithms, and Discrete Structures).

After being assigned to a specific set of courses, each ULA is responsible for holding office hours in a designated help desk room and assisting drop-in students. Such an opportunity for students to quickly seek help from their peers and seniors without prior appointments will help build a welcoming environment and a sense of community in addition to technical academic advancement.

It is important to note that the ULAs are not responsible for grading or course design in contrast to other initiatives[5]. This promotes a peer-learning atmosphere, eliminating the typical anxiety associated with seeking help from authoritative figures like TAs or instructors.

ULAs are also expected to participate in a one-hour weekly training session to discuss pedagogical strategies, rules and guidelines, effective communications, ethics, and their roles and responsibilities. By continuously improving the training, we provide and support the pursuit of other development opportunities that target the specific challenges of CS tutoring.

3 ASSESSMENT

This study examines the impact of the ULA program on students and ULAs by conducting a comprehensive analysis of academic records and self-reported experience surveys.

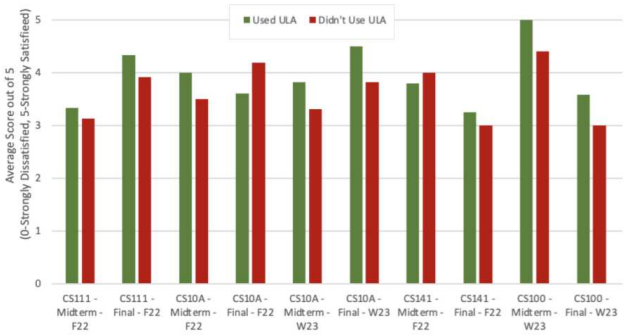


Figure 1: Average course satisfaction score reported by students enrolled in participating courses in various cohorts using a 0-5 Likert scale

3.1 Methodology

We deployed multiple surveys distributed to evaluate the program. These surveys can be categorized into three main groups:

- **End-of-Session Survey**
  Students receive a brief survey each time they attend a session to report their satisfaction, address concerns, and track progress throughout the quarter.
- **Midterm-Quarter/Quarter-End Course Survey**
  All the students enrolled in the courses covered by the ULA program, regardless of whether or not they have attended a ULA session, receive two surveys about their experience with the course, once after the midterm exam and once after the final exam. These surveys assist us in tracking whether participating in a ULA influences the overall course experience. The students report their contentment, sense of community, belonging, and progress using a 0-5 Likert scale.
- **ULA Experience Survey**
  The ULAs are requested to complete surveys at the start and end of each academic quarter. These surveys capture their perspectives on experiences, academic advancements, interactions with faculty and students, prospective interest in teaching roles, and more.

Beyond the data obtained from self-reported surveys, we explore the influence of the ULA program by examining the academic records of partaking students and evaluating course DFW (Drop, Fail, Withdraw) rates.

4 PRELIMINARY RESULTS

Since Fall 2022, assessing the ULA's impact has involved survey administration and student data collection. Our findings reveal that over 64% of surveyed students registered in the participating courses rated their satisfaction with the program at an average score of 4.6 out of 5 and acknowledged its favorable effect on their overall grades and assignments.

Moreover, as illustrated in Figure1, across most cohorts, students who utilize the ULA program in their courses consistently indicate superior course experiences compared to their non-participating peers and score the overall course higher.

We are currently engaged in the data collection process and analysis, focusing on gathering information from students' official academic records, analyzing DFW rates, and evaluating the ULA program's influence on participating students compared to those not involved.

5 CONTRIBUTION AND FUTURE WORK

We created and implemented a peer tutoring program with the objectives of improving student success in foundational CS courses, fostering a sense of community, particularly among underrepresented students in the early stages of their studies, and establishing an environment where talented and motivated students can transition into the role of a ULA, offering assistance to their fellow peers and possibly pursue teaching roles in the future. Furthermore, an in-lab ULA program is currently being tested in select courses to provide additional help for lab TAs. The program will also expand to other challenging courses in the near future.

ACKNOWLEDGMENTS

We are grateful to the Center of Inclusive Computing at Northeastern University for making this research possible[8].

REFERENCES

[1] S. Cheryan, V. C. Plaut, P. Davies, and C. M. Steele. 2009. Ambient belonging: how stereotypical cues impact gender participation in computer science. *Journal of Personality and Social Psychology* 97 (2009), 1045–1060. Issue 6. https://doi.org/10.1037/a0016239

[2] J. M. Cohoon, R. Shwalb, and L. Chen. 2003. Faculty turnover in cs departments. *ACM SIGCSE Bulletin* 35 (2003), 108–112. Issue 1. https://doi.org/10.1145/792548.611944

[3] Julia M Markel and Philip J Guo. 2021. Inside the mind of a cs undergraduate ta: A firsthand account of undergraduate peer tutoring in computer labs. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 502–508.

[4] Diba Mirza, Phillip T Conrad, Christian Lloyd, Ziad Matni, and Arthur Gatin. 2019. Undergraduate teaching assistants in computer science: a systematic literature review. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 31–40.

[5] Inna Pivkina. 2016. Peer learning assistants in undergraduate computer science courses. In *2016 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–4.

[6] Stuart Reges, John McGrory, and Jeff Smith. 1988. The effective use of undergraduates to staff large introductory CS courses. *ACM Sigcse Bulletin* 20, 1 (1988), 22–25.

[7] S. Tamachi, J. A. Giles, T. Dornan, and E. J. R. Hill. 2018. "you understand that whole big situation they're in": interpretative phenomenological analysis of peer-assisted learning. *BMC Medical Education* 18 (2018). Issue 1. https://doi.org/10.1186/s12909-018-1291-2

[8] Northeastern University. 2023. Center of Inclusive Computing, Northeastern University. https://cic.northeastern.edu/.

Generative AI as a Resource for Creativity in Computational Physics

Patti C. Hamerski
patti.hamerski@oregonstate.edu
Oregon State University
Corvallis, Oregon, USA

ABSTRACT

Generative artificial intelligence (gen-AI) has become ubiquitous in daily life, including classroom environments where students are using it to assist them on their coursework. Given the widespread use of this tool and the lack of knowledge over how it can support learning, there is a need for educators to have a framework for using it in the classroom and teaching their students usage strategies that are beneficial for learning. One pathway forward is through creativity, a process crucial for learning and also connected to the act of using gen-AI. This poster demonstrates the results of a study designed to provide an in-depth view on how creativity intersects with gen-AI usage in a computational physics course. In the course, students learn about computing tools during group-based, open-ended computational physics activities. Students are often tasked with using gen-AI to explore and help make decisions. The findings demonstrate a connection between using gen-AI and engaging in creative processes, and the implications point to strategies for supporting student usage of gen-AI.

CCS CONCEPTS

• **Social and professional topics** → **Computing literacy**; **Computational science and engineering education**.

KEYWORDS

generative AI; computational science; creativity; curriculum design

ACM Reference Format:
Patti C. Hamerski. 2024. Generative AI as a Resource for Creativity in Computational Physics. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635595

1 INTRODUCTION & BACKGROUND

Generative artificial intelligence (gen-AI) encompasses a variety of chatbot-style AI programs that can generate fluent text in response to prompts. It is becoming increasingly pervasive [2], and it is also becoming a tool for students to engage with computing problems in new ways that were never before possible [12]. It allows students to type a simple prompt in plain language, and provides relevant and meaningful content drafted up from the vast swath of information that it has access to. In computing classroom spaces, students are already using gen-AI as a resource to augment their programming practices [12]. The rapid emergence of this untested educational resource raises questions about how to support students in using gen-AI to benefit their learning process.

One path forward is to use gen-AI as an opportunity for building creativity [8]. Creativity is an important learning outcome in many contexts, and it also aligns with several aspects of computational thinking [6], another valuable learning outcome in computing and even non-computing disciplines. If creativity can be fostered through using gen-AI, then that can demonstrate a valuable role for AI to play in computing classrooms and in STEM education broadly.

Given this potential for gen-AI to be a resource for creativity in the classroom, I aim to investigate *how* this can be done. A computational physics course where students must primarily learn computing tools in service of physics disciplinary content can be an ideal place to carry out this study. Models for how students learn computational physics emphasize the importance of students use external resources to produce solutions [9]. This contextual choice also aligns with a conference report that framed computing as a foundational resource for learning physics [1]. With this framing towards supporting creativity through gen-AI in computational physics, my research questions are: **(RQ1)** How is the use of gen-AI connected to creative processes in a computational physics course? **(RQ2)** How can gen-AI serve as a useful resource to help students learn computational physics?

Part of what makes gen-AI a creative process is the writing of prompts [8]. Students often have to iteratively tailor the prompt that they write to get an output that is useful to them. The act of using gen-AI can be a creative process in this way, and it can also "spark" creativity by helping students organize and articulate ideas that are not yet fully fleshed out before consulting gen-AI [8]. The process of creativity in a computational physics setting can be conceptualized in terms of how students demonstrate fluency, flexibility, originality, and elaboration [11]. We use this breakdown to define creativity in terms of its observable components, which can also be helpful for conceptualizing what creativity can look like in the classroom.

2 CONTEXT

The context of this research is a sequence of computational physics courses at Oregon State University (OSU). The courses, Computational Physics Lab I, II, and III (PH 365/6/7, or PH 36X), are enrolled mainly by physics majors in their third year of post-secondary school. The students have varying computational backgrounds,

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635595

some even having no prior computing experience before PH 365, and the course sequence is meant to teach them the fundamentals of programming in Python and how to use computing as a tool for doing and learning about physics phenomena more deeply.

The types of physics problems that students engage with are often open-ended and result in students trying out different solutions before figuring out an approach that works for them. This design is similar to the act of "composing" ill-structured physics problems [10], which also parallels the potentially creative process involved in formulated prompts for ChatGPT. When students work on ill-structured or open-ended computing and physics problems, they often benefit from working in groups as well [5, 10]. With this in mind, the activities in the computational physics course sequence at OSU were designed to be well-suited for group work.

Designing a group-worthy activity [7] is important when students are expected to work together, because it helps them rely on each other in productive ways rather than out of feelings of obligation. A group-worthy task consists of working on open-ended, discipline-based computing problems with multiple entry points and opportunities to demonstrate competence [7]. For example, an activity in PH 365 asks students to write a Python program to estimate the power that could be produced from a roof designed to harvest energy from rain water. The activity is evaluated not only on the effectiveness of the program, but also how the group models the relevant physics, uses the model to draw insights, tracks assumptions, and analyzes implications.

3 METHODS

To investigate the impact of gen-AI in this context, I updated some of the activities in PH 36X, both to ensure the activities were group-worthy, but also to encourage and sometimes require students to use gen-AI to explore how it could be used as a tool in building a solution. The methodology chosen for this research was case study [3], because the detailed course context provided unique opportunities and constraints for how gen-AI could be used. Furthermore, case study can help build deep understanding of poorly understood processes. Given the novelty of gen-AI tools, it made sense to prioritize the construction of detailed knowledge that could be used to gain insight into possible processes of students creatively using this new resource.

Conducting a case study involves collecting several data sources to gain perspectives on the research phenomena from multiple angles. First, I took field notes to capture what students did and what choices they made about how to interact with ChatGPT and discuss its outputs. Second, I collected computing artifacts (i.e., completed activities) to examine how creativity translated into pieces of a solution that students chose to construct. Third, I assigned reflective tasks where students had to describe how they interpreted and framed their use of gen-AI, which helped gain insight into how students saw it as a resource in different ways.

To analyze these data, I employ discourse analysis [4], and apply codes from the creativity framework to construct themes around my research questions. I made this choice because these data sources capture different forms of student language and expression, and discourse analysis is well suited for a priori coding as well as building claims from different sources of language data.

4 RESULTS & DISCUSSION

Based on preliminary analysis, there are a few findings expected to be fleshed out from this research. One such finding is that students and student-prompted gen-AI outputs can jointly demonstrate creativity. An example can be seen in a field notes excerpt from PH 365: *"One student used ChatGPT during an in-class assignment. After not being able to fully understand its answer, she typed into it, 'break down every line of code like I'm a beginner,' and continued to explore the details with further prompts throughout class as she worked."*

In this excerpt, the student demonstrated *flexibility* by adapting to the constraints posed by an answer that did not make sense to her, and used ChatGPT as a tool for *elaboration* in terms of how it provided further details about an incomplete explanation. This instance suggests that aspects of creativity can come from students, in response to AI (like with the flexibility), but also can come from AI in response to students-crafted prompts (like with the elaboration).

Further potential findings from preliminary analysis are as follows. First, students can practice creativity when using gen-AI as a resource in class. Second, gen-AI can be an effective resource to help students create computing solutions to physics problems. Third, gen-AI can support the development of creativity on a group level because of how students engage with it by tailoring prompts, interacting with its outputs, and negotiating how to use it in their solution. I look forward to presenting more fleshed out findings and contextual details from this ongoing research in my poster.

REFERENCES

[1] Marcos D. Caballero, Kathi Fisler, Robert C. Hilborn, Colleen Megowan Romanowicz, and Rebecca Vieyra. 2020. *Advancing Interdisciplinary Integration of Computational Thinking in Science.* The American Association of Physics Teachers, College Park.

[2] Grant Cooper. 2023. Examining Science Education in ChatGPT: An Exploratory Study of Generative Artificial Intelligence. *Journal of Science Education and Technology* 32, 3 (2023), 444–452. https://doi.org/10.1007/s10956-023-10039-y

[3] Kathleen M. Eisenhardt. 1989. Building theories from case study research. *The Academy of Management Review* 14, 4 (1989), 532–550.

[4] James Paul Gee. 2014. *How to do Discourse Analysis: A Toolkit.* Routledge, New York, NY.

[5] Patti C. Hamerski, Devin Silvia, and Marcos D. Caballero. 2022. Exploring Self-efficacy in Data Science. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education.* Dublin, Ireland.

[6] Rotem Israel-Fishelson and Arnon Hershkovitz. 2022. Studying interrelations of computational thinking and creativity: A scoping review (2011–2020). *Computers & Education* 176 (2022), 104353. https://doi.org/10.1016/j.compedu.2021.104353

[7] Rachel A. Lotan. 2003. Group-Worthy Tasks. *Educational Leadership* (2003), 72–75.

[8] Rebecca Marrone, Victoria Taddeo, and Gillian Hill. 2022. Creativity and Artificial Intelligence—A Student Perspective. *Journal of Intelligence* 10, 3 (2022), 65. https://doi.org/10.3390/jintelligence10030065 AI "can spark creativity".

[9] A. M. Phillips, E. J. Gouvea, B. E. Gravel, P.-H. Beachemin, and T. J. Atherton. 2023. Physicality, modeling, and agency in a computational physics class. *Physical Review Physics Education Research* 19, 1 (2023), 010121. https://doi.org/10.1103/physrevphyseducres.19.010121

[10] Javier Pulgar, Valentina Fahler, and Alexis Spina. 2021. Investigating how university students collaborate to compose physics problems through structured tasks. *Physical Review Physics Education Research* 17, 1 (2021), 010120. https://doi.org/10.1103/physrevphyseducres.17.010120

[11] Muhammad Tawil and Ahmad Dahlan. 2017. Developing Students' Creativity through Computer Simulation Based Learning in Quantum Physics Learning. *International Journal of Environmental & Science Education* 12, 8 (2017), 1829–1845.

[12] Ramazan Yilmaz and Fatma Gizem Karaoglan Yilmaz. 2023. The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and Education: Artificial Intelligence* 4 (2023), 100147. https://doi.org/10.1016/j.caeai.2023.100147

A Systematic Literature Mapping of COVID-19 Papers in Computer Science Education

Brian Harrington
Sharon Alex
Leon Lee
brian.harrington@utoronto.ca
sharon.alex@mail.utoronto.ca
leonlee.lee@mail.utoronto.ca
University of Toronto Scarborough
Toronto, ON, Canada

Colin Lin
Zixiao Ren
Youxin Tan
colin.lin@mail.utoronto.ca
zixiao.ren@mail.utoronto.ca
youxin.tan@mail.utoronto.ca
University of Toronto Scarborough
Toronto, ON, Canada

Conroy Trinh
Shengsong Xu
Austin Yang
conroy.trinh@mail.utoronto.ca
shengsong.xu@mail.utoronto.ca
au.yang@mail.utoronto.ca
University of Toronto Scarborough
Toronto, ON, Canada

ABSTRACT

The COVID-19 pandemic caused many institutions to radically alter the way in which they delivered and evaluated their educational models and frameworks. In the subsequent 3 years, a great deal of research has been conducted in the CS Education community as to the changes that were made and how they impacted learners, educators, and institutions.

Building on our existing framework for an interim literature map, this work seeks to provide a systematic and complete mapping of the relevant literature, providing both a tool for practitioners to easily find and synthesize literature, and the basis for researchers to perform a more detailed literature review.

This work represents a systematic search, categorization, refinement and validation of a literature mapping on three axes: Intervention, Measurement, and Population. An accompanying bibliography and mapping data file is also available for download.

ACM Reference Format:
Brian Harrington, Sharon Alex, Leon Lee, Colin Lin, Zixiao Ren, Youxin Tan, Conroy Trinh, Shengsong Xu, and Austin Yang. 2024. A Systematic Literature Mapping of COVID-19 Papers in Computer Science Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635538

EXTENDING INTERIM MAPPINGS

This work builds upon the protocol established in Harrington et al. [1], in which we performed a systematic search of papers in the first 2 years following the onset of the pandemic.

In the earlier pilot work, we established a search and inclusion protocol, as well as a baseline framework for axis of analysis and a process of categorization.

While it is important to acknowledge that COVID-19 is still an active problem in much of the world. Most of the large-scale lockdowns and quarantines that impacted classrooms have now abated, and research performed during this transition has now had three years for publication. Therefore, while our previous work represented an interim view of the pandemic's impact on CS education. This work is designed to be a summative analysis.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635538

MAPPING PROTOCOL AND REFINEMENT

Following the protocol for literature mapping set forth in Kitchenham and Charters [2], and using parameters and processes validated in our previous mapping project[1]. On June 6, 2023 the query **covid OR coronavirus OR pandemic** was run on the ACM Digital Library, limiting the scope to events sponsored by the ACM SIGCSE. This returned an initial database of 425 total articles.

We applied the following inclusion criteria:

- must be a research article of more than 1 page
- must detail some mitigation strategy or impact of the covid pandemic
- performing a systematic measurement on some aspect of CS education.

Using a double evaluator model (Kohen's Kappa: 0.92), we found a total of 55 papers which meet the criteria.

All papers were coded on three dimensions

- Intervention: remote teaching, remote assessment, remote collaboration, activities/community, general impact
- Measurement: grades, non grade performance, attendance/retention, affect/perceptions, mental health
- Population: K-12, university, introductory, educators, gender, race

using a double evaluator model (Cohen's Kappa: 0.78), with discrepancies resolved through group discussion or third party mediation.

PAPER DENSITY BY CATEGORY

As can be seen in the heat-maps in Tables 1-3, there is a great deal of variation in the density of research by area. While 21 studies looked at the perceptions of remote teaching, only 1 study looked at the impact of remote-assessment on non-grade performance. And while 17 of the 55 papers specifically dealt with CS0/CS1 students, only 4 papers looked at the impact upon educators, and 9 papers differentiated results based on gender and/or race.

This imbalance in study density means that while some areas have replicated results and basis for comparison, other areas remain woefully under-studied.

	Grades	Performance	Participation	Perceptions	Mental Health
Remote Teaching	5	0	5	21	1
Remote Assessment	11	1	3	11	1
Remote Work	4	0	3	16	2
Pandemic General	2	1	2	8	3
Virtual Event	1	2	1	4	0

Table 1: Interventions vs Measurements

	K-12	Undergrad-uate	Introduc-tory	Educators	Gender	Race
Grades	3	12	7	0	0	0
Performance	2	2	2	1	0	0
Participation	3	6	2	0	1	1
Perceptions	10	28	11	3	5	5
Mental Health	0	5	3	0	1	1

Table 2: Measurements vs Populations

	Remote Teaching	Remote Assessment	Remote Work	Pandemic General	Virtual Event
K-12	9	2	4	1	5
Undergraduate	17	13	16	11	0
Introductory	4	8	5	4	0
Educators	2	0	0	1	2
Gender	2	0	2	2	1
Race	3	0	1	3	1

Table 3: Populations vs Interventions

The unplanned and necessarily spontaneous nature of pandemic research obviously lent itself to certain types of studies. It is understandable that perceptions and grades are the most common forms of measurements as they are often already in place, and do not require a long planning phase. While studies involving K-12 students, gender, race, or mental health may well have been impossible in many situations due to the time constraints of developing adequate instruments and securing ethical clearance. However, these discrepancies are also indicative of the balance of focus on areas that are already prevalent within the community, and may serve as a window into what sorts of factors we as a research community are prepared to study on short notice.

While we will never again (hopefully) have the opportunity to study the impacts of this specific pandemic on our students, this work may serve as a guide to what areas are being under-studied in general, and as a preparatory guide for future research plans.

FUTURE WORK

We hope that this literature map will serve as a resource to researchers, both as a 'snapshot' of pandemic research, and a guide to future development. We also plan to use this mapping as the basis for a more thorough literature review, focused specifically on meta-analysis of the variance within interventions and measurements.

RESOURCES

The complete mapping is available as either a PDF or downloadable CSV with accompanying bibliography file at

https://github.com/BrianHarringtonUTSC/SIGCSE2024CovidLitMap

in order to allow researchers to view the mappings or analyze the data themselves.

The poster will detail not only the process of completing the map, but also an analysis and the findings of the researchers, including areas of high/low density of research, systematic trends, and patterns with respect to time.

REFERENCES

[1] Brian Harrington, Aditya Kulkarni, Zixiao Ren, Conroy Trinh, Raha Gharadaghi, Thezyrie Amarouche, Ansh Aneel, Anand Karki, Seemin Syed, and David Yue. 2023. Finding and Categorizing COVID-19 Papers in CS Education. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1342.

[2] Barbara Kitchenham and Stuart M. Charters. 2007. *Guidelines for performing systematic literature reviews in software engineering.* Technical Report. Technical report, ver. 2.3 ebse technical report. ebse.

Alternative Evaluation in CS Education Research: A Systematic Literature Map

Brian Harrington
Thezyrie Amarouche
Andrew Aucie
Shreeansha Bhattarai
University of Toronto Scarborough
Toronto, ON, Canada

Yuxin Chen
Raha Gharadaghi
Kaiyi Huang
Jin Jiang
Linda Lo
University of Toronto Scarborough
Toronto, ON, Canada

Maliha Lodi
Rohita Nalluri
Fawaz Omidiya
Anagha Vadarevu
University of Toronto Scarborough
Toronto, ON, Canada

ABSTRACT

There is a growing movement in the education community to re-evaluate grading practices and find ways of evaluating students that are more flexible, equitable, and focused on student growth. A wide variety of alternative evaluation systems have been developed, including mastery grading, specifications grading, contract grading and ungrading.

In this work, we perform a systematic literature search to find studies of alternative evaluation models in computer science education. A total of 37 papers were analyzed and coded on three metrics: Intervention, Evaluation, and Population.

This map represents a tool for educators to find and evaluate research on alternative evaluation systems, a resource for researchers to identify areas of future research, and a foundation for a future detailed literature analysis. The complete bibliography and literature map are available for download.

ACM Reference Format:
Brian Harrington, Thezyrie Amarouche, Andrew Aucie, Shreeansha Bhattarai, Yuxin Chen, Raha Gharadaghi, Kaiyi Huang, Jin Jiang, Linda Lo, Maliha Lodi, Rohita Nalluri, Fawaz Omidiya, and Anagha Vadarevu. 2024. Alternative Evaluation in CS Education Research: A Systematic Literature Map. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635537

MOTIVATION FOR A LITERATURE MAP

There is a growing movement to re-think our models of evaluation across academia to promote equity and student growth [1, 2]. This movement has been gaining momentum in the field of computer science education (over 50% of the documents retrieved by our initial query were published within the past 5 years). As shown in [3], there is no consensus on the terminology or standards for these practices within the literature.

As Kitchenham and Charters [5] states, a systematic literature map "allows for the identification of evidence clusters and evidence deserts to direct the focus of future systematic reviews and to identify areas for more primary studies to be conducted"

We feel that this systematic mapping exercise will provide both a resource that will be of value to the community (the full map and .bib file is available for download), as well as the foundation for future work into a systematic review of the sub-topics identified.

SEARCH AND MAPPING PROTOCOL

Following the literature mapping protocols set forth in Kitchenham and Charters [5], and prior literature maps in CS education [4], on June 6, 2023 the query **("mastery grading" OR "alternative grading" OR "specification grading" OR "specifications grading" OR "mastery learning" OR "specs grading" OR "grading for equity" OR "equitable grading" OR "ungrading OR un-grading OR "contract grading") AND (programming OR computer science OR coding)** was run on the ACM Digital Library, Scopus, Science Direct and IEEE Xplore. An initial set of 376 papers were evaluated against inclusion criteria resulting in a set of 37 peer-reviewed research papers longer than 1 page evaluating some aspect of alternative evaluation on a population relevant to computer science education.

All papers were coded on three axes: Feature Being Evaluated (Binary Grading, Bucketed Grading, Mastery Learning, Resubmissions, Minimum Grades, Student Input on Grades), Evaluation Metric (Grades/Course Outcomes, Engagement, Student Preferences, Psychological Factors, Teaching Workload, Student Workload), and Population (K-12, Undergraduate, Educators, Introductory Learners, Gender, Race, Accessibility). After an initial multi-rater clustering showed poor inter-rater reliability (Cohen's Kappa = 0.5715), a second pass coding with pre-chosen categories produced good inter-rater reliability (Cohen's Kappa = 0.7860), with all discrepancies resolved by discussion and mutual agreement.

MAPPING AXES & CATEGORIES

As is standard protocol following Kitchenham and Charters [5], the first stage of the mapping process was to choose axes for mapping. The agreed upon axes were: the intervention being studied, the measurement used to evaluate the efficacy of the intervention, and the population upon which the study was conducted.

After an initial clustering, the following categories were identified within the clusters:

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635537

Intervention being studied:

- Binary Grading: Any study in which course components are simply graded as “complete" vs “incomplete" (or in the language of mastery based evaluation “mastery demonstrated" vs “mastery not yet demonstrated")
- Bucketed Grading: Similar to binary grading, but allowing for a small number of gradations of competency. Often evaluated as “excellent", “satisfactory", or ‘not yet demonstrated"
- Mastery Learning: While the terminology has been used in other ways, in this context it refers to evaluation being based on how far in a series of gated activities the student progresses, with successive stages only being open after a certain level of mastery is demonstrated
- Re-submission Without Penalty: Studies where students resubmit or re-try work an unlimited (or quasi-unlimited) number of times, or allowing demonstration of any course component at any time during the course duration
- Minimum Grades: Courses which assign a non-zero minimum grade, even for incomplete or non-submitted work
- Student Input on Grades: Studies where students are asked to decide upon, or lobby for the grade they feel they deserve, or where the grade, rubric or weighting is decided in consultation with students

Evaluation Metric:

- Grades/Course Outcomes: Any component of a course evaluation on a current or subsequent course
- Engagement: Any evaluation based on participation or attendance rates, including drop/withdrawal rates
- Student Preferences: Evaluation where students are asked their preferences or opinions on any course component, or their feelings towards the course, program or subject of study
- Psychological Factors: Any evaluation of the students’ mood, stress, sense of belonging, mindset, or other factors not accommodated for in preferences
- Teaching Workload: Any evaluation of the amount of work to set up or maintain the alternate evaluation model
- Student Workload: Any evaluation of the amount of work completed by students, whether mandatory or voluntary

Population:

- K-12/Undergraduate: When identified, the current level of education at which the course is being offered
- Educators: Any evaluation of the impact on the course instructors or other education professionals
- Introductory Students: CS0/CS1 or any context in which students are encountering programming for the first time
- Gender/Race: Any study in which results are differentiated by gender/race
- Accessibility: Any study in which accessibility issues or considerations for neurodivergent students is evaluated

PRELIMINARY FINDINGS

While this work is focused on a literature mapping, and not a full literature review; the process has uncovered several important elements within the literature:

- There is no consistency of terminology or best practices within the literature. While many papers referred to “contract grading" or “specifications based evaluation" as though they were a well understood and organized evaluation system, the map clearly shows that papers using the same terminology have different components, and some papers which share components do not use the same terminology.
- Implementation details vary greatly within components. Even where studies use the same terminology, or refer to the same reasoning for a particular element of alternative grading, the practices are often quite dissimilar. As one example demonstrates, while 20 of the 37 papers proposed to allow unlimited attempts to complete work (with many of them referring to Feldman [2] and the principle that students should be evaluated on “what they learned, not when they learned it"), in practice, every single study had to enforce some limits on re-submissions or re-attempts, but no two studies had identical models, making comparison difficult.
- There is a major imbalance in study density across categories. As can be plainly seen on the heat-maps presented on the poster, or on the full literature map available for download, there are some areas that are well studied (e.g., the intersection of “Undergraduate" and “Grades", or “Mastery Learning" and “Preferences" having more than 20 papers), and others that are desperately in need of evaluation (e.g., only 4 total papers taking into account the impact on educators, or only 2 total papers evaluating the impact by gender or race). While this is understandable as the research area is only recently growing in popularity, it points to both an important need to formalize replication in repeated areas, and begin programs of study in under-served categories.

POSTER & DOWNLOADABLE MAP

The complete mapping is available as either a PDF or downloadable CSV with accompanying bibliography file at:
https://github.com/BrianHarringtonUTSC/SIGCSE2024AltGradingLitMap

The poster will detail the mapping protocol and process, as well as the findings of relative densities of papers as shown in the literature mapping to indicate areas of existing research and areas of need for future work. While a complete map cannot be included on the poster itself, a heat-map will indicate the areas of concentration and scarcity within the full literature map.

REFERENCES

[1] David Clark and Robert Talbert. 2023. *Grading for Growth: A Guide to Alternative Grading Practices that Promote Authentic Learning and Student Engagement in Higher Education.* Taylor & Francis.
[2] Joe Feldman. 2018. *Grading for equity: What it is, why it matters, and how it can transform schools and classrooms.* Corwin Press.
[3] Brian Harrington, Abdalaziz Galal, Rohita Nalluri, Faiza Nasiha, and Anagha Vadarevu. 2024. Specifications and Contract Grading in Computer Science Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024).*
[4] Brian Harrington, Aditya Kulkarni, Zixiao Ren, Conroy Trinh, Raha Gharadaghi, Thezyrie Amarouche, Ansh Aneel, Anand Karki, Seemin Syed, and David Yue. 2023. Finding and Categorizing COVID-19 Papers in CS Education. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE 2023).*
[5] Barbara Kitchenham and Stuart M. Charters. 2007. *Guidelines for performing systematic literature reviews in software engineering.* Technical Report. Technical report, ver. 2.3 ebse technical report. ebse.

Towards Reuse of Resources and Replication of Results to Improve Instruction on Refactoring Conditionals for Novice Programmers

Cecily Heiner
Computer Systems Engineering Technology
Oregon Institute of Technology
Klamath Falls Oregon USA
cecily.heiner@oit.edu

Dana Nelson
Computer Systems Engineering Technology
Oregon Institute of Technology
Klamath Falls Oregon USA
dana.nelson@oit.edu

ABSTRACT

Creating resources such as tools, curricula, and assessments is an important aspect of educational work, but dissemination and adoption of these resources is often hard. Surprisingly few papers have been written about reusing resources and replicating corresponding research results for an individual lesson of 30-60 minutes. This poster describes our reuse of an instructional resource on conditional refactoring and our attempts to partially replicate the results reported in the paper that originally described the resource. We describe what parts were easy to replicate as well as difficulties. We hope to help others who wish to promote adoption of new instructional materials and replication of their research results.

ACM Reference Format

Cecily Heiner & Dana Nelson. 2024. Towards Reuse of Resources and Replication of Results to Improve Instruction on Refactoring Conditionals for Novice Programmers. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA. ACM, New York, NY USA.* https://doi.org/10.1145/3626253.3635626

1 BACKGROUND

Although most in the Special Interest Group Computer Science Education(SIGCSE) community would probably generally agree that novice programmers should learn something about input, output, loops, and arrays, there are many other educational design decisions that remain highly controversial, including but not limited to the human language, programming language, toolset, approach(objects first or last, with or without a graphical user interface, etc), platform, curriculum, textbooks, and others. The diversity of choices can make it difficult to replicate and reproduce research findings relating to instruction in the classroom. Replication and reproduction are important aspects of the scientific process that are often curiously absent in this research community.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03. https://doi.org/10.1145/3626253.3635626

This poster describes our initial attempt to reuse a resource and partially replicate a result previously described in an Innovation and Technology in Computer Science Education (ITICSE) paper on conditional refactoring [1]. We benefit from using the same human language (English) and programming language(C++), although we are on different sides of the world, and like the paper that we aim to replicate, we attempt to analyze the student data on a slightly deeper level than correct/incorrect, investigating instead the number of lines of code and presence of if tokens. Our results approximately mirror the study we are attempting to replicate.

2 PRIOR WORK

We divide our discussion of prior work into two main strands: prior work on conditionals in novice programming and replication of results using a classic problem such as the so-called rainfall problem [2]. Early work by Soloway and Spohrer describes how pre-programming knowledge can lead to bugs [3]. Pea explains how usage of the word if in English can be different than the use of the same term in a programming language [4]. Guzdial notes that some languages such as ProLog eliminate the need for an explicit condition as it is written in C based programming languages [5]. Kwon focused on plans, but includes several extended examples of prompts and students responses to problems focused on programming with conditionals [6]. Recent work on conditionals by novice programmers has been completed by Cruz e.g.[1], and replicating [1] is the primary topic of this paper. Many groups have attempted to replicate results of success/failure rates on a few popular programming problems including the so-called rainfall problem, and an extensive paper considering many studies of this problem in various contexts can be found in [2]. These studies primarily deal with assessment on a given problem, and generally do not include relevant instruction on a given topic.

3 METHODOLOGY

We aimed to replicate the results in this 2022 ITICSE paper on conditional refactoring[1]. The study described there uses a resource that outlines four rules to reduce the length and the number of if statements in short snippets of code. The resource also provides nine worked examples illustrating the rules and

three sample exercises for students to practice applying the rules. We were able to obtain the resource as an instructional worksheet in pdf form complete with worked examples as well as the three sample exercises. We also obtained an Excel spreadsheet used for analysis of one of the three sample exercises, but we did not obtain analysis tools for the other two sample exercises. We did not obtain prompts for the long-term follow-up programming problems, so we did not attempt to replicate that part of the experiment. After obtaining the resource, we obtained IRB approval to run the experiment in two sections of our introductory C/C++ programming class taught by two different instructors; we note that obtaining IRB approval even for a study that should obviously qualify for expedited review, can be a hurdle in our institution. We collected data for the 3 sample exercises along with the median problem in the form of a .cpp file through our Learning Management System(Canvas) Each instructor removed any data from non-consenting students from the other instructor's dataset, leaving us with 12 students in one section and 13 in the other. We then proceeded to manually count the use of if statements and count the lines of code, excluding newlines and comments. We did not verify accuracy, and we did not provide feedback on the sample exercises.

4 RESULTS

We use a refactoring resource with 3 exercises. For each exercise, we count the if statements and the number of lines of code in the function body excluding comments and new lines. Exercise A has 7 lines of code and a single if-else statement, and it can be factored into a single line of code with no if statements. Exercise B has 6 lines of code and a nested if statement, and it can be refactored into 4 lines of code with a single if statement. Exercise C has 18 lines of code and 5 if statements. We note that the original paper is primarily focused on the resource itself, and much of the quantitative data is buried in prose. We did not have access to the original raw data, so numbers reported for the original paper generally came from that paper. In some cases, the prose suggested some ambiguity, e.g. "partial refactoring included the removal of the conditional statement or the removal of the variable"; however, the paper did not disaggregate these two options, so we simply report it at as + in the middle of the second row. We note significant differences in the number of students that were able to reduce Exercise A to a single line of code, even though the resource clearly indicated "Hint: it is possible to rewrite this function using just a single line of code." We speculate that our students may not have seen this hint, or they may have been more interested in removing if statements than eliminating variables, or they may not have been motivated, since there was no academic credit tied to participation in this experiment. In both the original paper and the replication, the vast majority of students were able to factor out the nested if statement. The original paper did not report an average number of lines of code for this problem. For the third exercises, we estimate that both cohorts used approximately 3 if statements. The number reported in the paper is 2.8, and we report numbers to the 10ths place for consistency, but we argue that counting numbers would be more appropriate for reporting lines of code in general. The original paper claims that exercise 3 has 24 lines of code; this appears to include the function header, closing curly brace, and all comments; this paper would claim that Exercise 3 has 18 lines of code, excluding the function header, closing curly, and comments. Unfortunately, the original paper was not super consistent in style, sometimes putting the opening curly on the same line as the function header and other times putting it on its own line. Counts of lines of code per exercise are equally obscure, sometimes reporting lines of code in the function as lines of code in the function body and other times reporting it as lines of code in the entire function, including the header and closing curly as well as comments.

Exercise A	Original Paper	Replication
No If's present	43%+	28%
Single line of code	43%	4%
Exercise B	Original Paper	Replication
Single If used	79%	96%
Lines of Code	Not Reported	4
Exercise C	Original Paper	Replication
If's Present	2.8	2.9
Lines of Code	15.7	12.0

5 LIMITATIONS, FUTURE WORK, AND CONCLUSIONS

Simple, easy-to-use generic tools for analysis of artifacts from novice programmers such as token detectors and line counters might be broadly useful for a variety of languages; ideally we would love to see a toolset that can take a folder of .cpp files and generate a table with things like token counts, line counts, and other basic counts with one column per count, one row per file. Additionally, studies of individual lessons and artifacts are easier to replicate than studies of entire curricula.

REFERENCES

[1] Izu, Cruz, Paul Denny, and Sayoni Roy. A Resource to Support Novices Refactoring Conditional Statements. *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education* Vol. 1. 2022.

[2] Seppälä, Otto, et al. "Do we know how difficult the rainfall problem is?." *Proceedings of the 15th Koli Calling Conference on Computing Education Research*. 2015.

[3] Soloway, Elliot, and James C. Spohrer, eds. Studying the novice programmer. Psychology Press, 2013..

[4] Pea, Roy D. "Language-independent conceptual "bugs" in novice programming." Journal of educational computing research 2.1 (1986): 25-36.

[5] Fincher, Sally, and Marian Petre. "Programming environments for novices." Computer science education research. Taylor & Francis, 2005. 137-164.

[6] Kwon, Kyungbin. "Novice Programmer's Misconception of Programming Reflected on Problem-Solving Plans." International Journal of Computer Science Education in Schools 1.4 (2017): n4.

Support Patterns in Classrooms Implementing a Computer Science and Physical Computing Curriculum

Nga Hoang
School of Education
University of Colorado Boulder
Boulder, CO, US
nga.hoang-1@colorado.edu

Jeffrey B. Bush
Institute of Cognitive Science
University of Colorado Boulder
Boulder, CO, US
Jeffrey.Bush@colorado.edu

Indrani Dey
Educational Psychology
University of Wisconsin-Madison
Madison, WI, US
idey2@wisc.edu

ABSTRACT

Teacher support is pivotal for promoting collaborative learning. Students often face challenges in self-regulating their learning within small group settings, especially when presented with open-ended learning opportunities. Analyzing 542 moments of support in small group work with the MOSAIC protocol during a unit on physical computing suggests a relationship between support type and activity design. Students showed more initiative in seeking support in more complex tasks. Teacher support focused on small groups rather than the entire class during complex or open-ended activities. Support focus on the direction of the task is the most common type across all activities. Support for collaboration was most frequent in the activity requiring students to coordinate knowledge developed individually. This analysis provides insights into potential areas for enhancing student support in a computer science classroom, particularly for collaboration.

ACM Reference format:
Nga Hoang, Jeffrey B. Bush, and Indrani Dey. 2024. Support Patterns in Classrooms Implementing a Computer Science and Physical Computing Curriculum**.** In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635574

INTRODUCTION AND BACKGROUND

Promoting classroom collaboration offers significant benefits, including improved learning outcomes [1], fostering inclusivity and equity in the classroom, with a particular benefit to female students [2], and the development of employer-valued skills [3]. However, students often encounter challenges controlling their learning during small group work [4]. Simply grouping students does not automatically result in productive collaboration [5]. "Smart" groups – where the expertise to solve problems is present, can often fail [6].

Before introducing strategies to enhance collaboration in classrooms, understanding existing dynamics is crucial. Our study examines classrooms implementing the Sensor Immersion (SI) unit, a curriculum designed to enhance collaboration, to answer three key questions: (1) Who initiates support, teachers or students, in various classroom activities? (2) Who are the recipients, the entire class, specific groups, or individual students, in various classroom activities? (3) How do support patterns, focusing on task completion and collaboration, vary across different types of activities?

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635574

METHOD

This study was conducted within a research-practice partnership between the Institute for Student and AI Teaming and two rural and suburban districts in the Western United States. The instructional context is the SI unit, a CS curriculum that focuses on teaching students to program, wire, and use sensors and significantly emphasizes developing their collaborative skills through diverse small-group activities:

1. *Program and Wiring Activity*: Students work with physical sensor systems, including environmental, sound, and soil sensors.
2. *Card-Sort Activity*: Students discuss to identify effective explanatory science models and their characteristics.
3. *Classroom Norms Discussion*: Students participate in a meta-collaborative activity to brainstorm community agreements to enhance their collaboration.
4. *Jigsaw Activity*: Individual students gain expertise in one of three sensors. They then work in groups to share their sensor knowledge with others, promoting positive interdependence.

Students worked in small groups of 2-4, asking for and receiving support from the teacher as needed. Videos of these interactions were captured using iPads and Yeti microphones. 205 videos, approximately 21 hours of small group work, were collected from four classrooms.

In the videos, we identified *support moments* as instances where individuals outside the recorded group provided guidance or instructions to group members. We then applied the Moment of Support Analysis in Classrooms (MOSAIC) coding protocol to analyze these support moments, including identifying initiators, recipients and categorizing the support into twelve specific categories [7]. There were 542 moments analyzed. We used Krippendorff's Alpha to assess coding reliability [8]. The result for the final coding was an acceptable 0.69.

FINDINGS

Among 542 identified moments, 65 support moments were associated with the card sort activity, 30 with classroom norms discussions, 356 with programming and wiring, and 91 with the jigsaw activity.

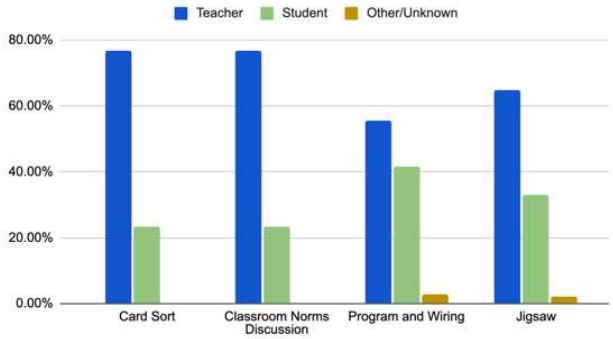


Figure 1. Initiators of support across all activities

Though the teacher most frequently initiated support across all activities, students more proactively initiated support in the program and wiring activity, the most complex of the tasks. This accounted for 41.5% in program and wiring compared to 23%-33% in the other activities (Figure 1).

There were variations in whom the teacher provided support depending on the activity (Figure 2). In the card sort and jigsaw activities, the teacher typically addressed the entire class (52% and 46%, respectively), while in classroom norms discussions and programming and wiring activities, the teacher primarily engaged with specific groups (57% and 48%, respectively). The task design, which is open-ended in classroom norm discussions and a complex task with context-based problems in programming and wiring activities, might contribute to this result.

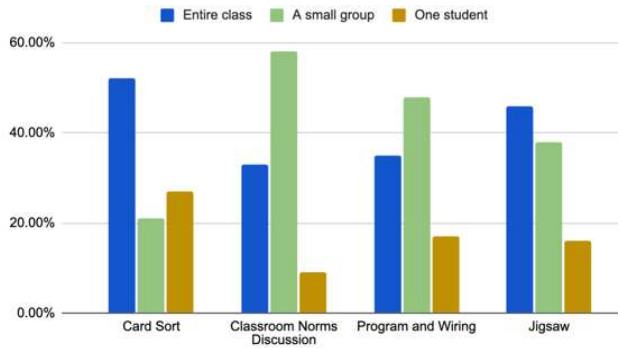


Figure 2. Support recipients across all activities

Support across activities focused on providing directions about assignments was high for all activities, ranging from 58% in model card sorting to 73% in programming and wiring. Collaboration-focused support was rare across all activities, with a notable variation. It was highest at 18% in the jigsaw activity but was six times less, at 3%, in discussions of classroom norms (Figure 3). The jigsaw activity required students to share knowledge developed individually with peers to complete a task where they had to apply that knowledge.

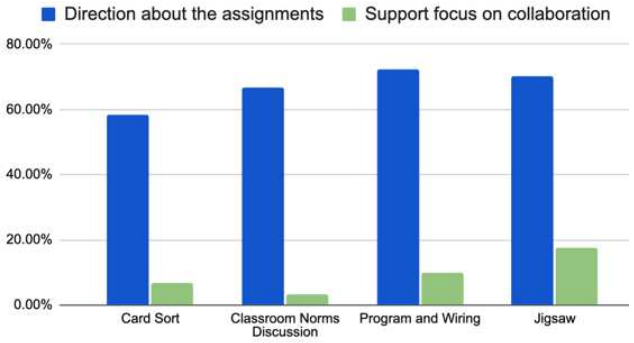


Figure 3. Type of support across all activities

DISCUSSION

There appears to be a link between activity design and the requested or provided support type. Teachers often initiated support, but students sought help more actively during more complex tasks. Collaboration support was most evident in tasks that made high demands on students' collaboration skills; that is, in the jigsaw activity. However, support for collaboration was less common than task-focused support in all activities.

Future research will explore the effects of teacher- or student-initiated support and to whom this support is provided. We also aim to understand why support for collaboration is limited, examining whether activity designs adequately promote teamwork or if teachers and students underestimate the need for such support.

REFERENCES

[1] Bower, M., & Richards, D. 2006. Collaborative learning: Some possibilities and limitations for students and teachers. *23rd Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education: Who's Learning* (pp. 79-89).

[2] Werner, L. L., Hanks, B., & McDowell, C. 2004. Pair-programming helps female computer science students. *Journal on Educational Resources in Computing*, 4(1), 4. https://doi.org/10.1145/1060071.1060075.

[3] Yoder, N., Atwell, M. N., Godek, D., Dusenbury, L., Bridgeland, J. M., & Weissberg, R. 2020. Preparing Youth for the Workforce of Tomorrow: Cultivating the Social and Emotional Skills Employers Demand. SEL for Workforce Development. *Collaborative for Academic, Social, and Emotional Learning*

[4] Molenaar, I., Van Boxtel, C. A. M., & Sleegers, P. J. C. 2010. The effects of scaffolding metacognitive activities in small groups. *Computers in Human Behavior*, 26(6), 1727–1738. https://doi.org/10.1016/j.chb.2010.06.022.

[5] Kreijns, K., Kirschner, P. A., & Jochems, W. 2003. Identifying the pitfalls for social interaction in computer-supported collaborative learning environments: a review of the research. *Computers in human behavior*, 19(3), 335-353.

[6] Barron, B. 2003. When Smart Groups Fail. *Journal of the Learning Sciences*, 12(3), 307–359. https://doi.org/10.1207/S15327809JLS1203_1

[7] Dey, I., Bush, J.B., Hoang, N. 2024. Analyzing Support Moments During Small Group Work. Poster Presentation. *American Educational Research Association*, Philadelphia, PA, USA.

[8] Hayes, A. F., & Krippendorff, K. 2007. Answering the Call for a Standard Reliability Measure for Coding Data. *Communication Methods and Measures*, 1(1), 77–89. https://doi.org/10.1080/19312450709336664.

Incorporating Retakes in a Robotics Class with Labs

Janine Hoelscher
jhoelsch@cs.unc.edu
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA

Bryce Ikeda
bikeda@cs.unc.edu
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA

Daniel Szafir
daniel.szafir@cs.unc.edu
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA

Ron Alterovitz
ron@cs.unc.edu
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA

ABSTRACT

Lab assignments, in which students build and program robots to accomplish tasks in various environments, are a central component in many undergraduate robotics classes. Such activities require that students operationalize concepts learned in class. However, physical robots are prone to uncertain real-world behavior, making debugging challenging and causing many students to feel stressed about being graded based on their robot's performance. Therefore, we incorporated retakes into our undergraduate robotics class, allowing students to learn from their mistakes and master class content while improving their robots. Initial results show that students widely embrace retakes, use the opportunity to improve, and feel less stressed about the assignments.

ACM Reference Format:
Janine Hoelscher, Bryce Ikeda, Daniel Szafir, and Ron Alterovitz. 2024. Incorporating Retakes in a Robotics Class with Labs. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635485

1 PROBLEM AND MOTIVATION

Designing hardware and software components for robots can be a rewarding experience but also very challenging. In our undergraduate junior/senior level robotics class, students build and program wheeled Lego robots in four labs throughout the semester. In groups of two or three that stayed the same throughout the semester, students incrementally build and improve autonomous robots that navigate around obstacles and move to goal locations, as shown in Fig. 1. These lab activities are designed to let students operationalize concepts covered in class lectures and are common in many undergraduate robotics classes (see [2] for further details). While the class is popular with students, the robotics labs cause some students stress, as part of their grade depends on live demonstrations of their robots performing an assigned task in an environment with never-before-seen obstacles. These high-stakes lab demonstrations serve as an evaluation and do not maximize student learning.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635485

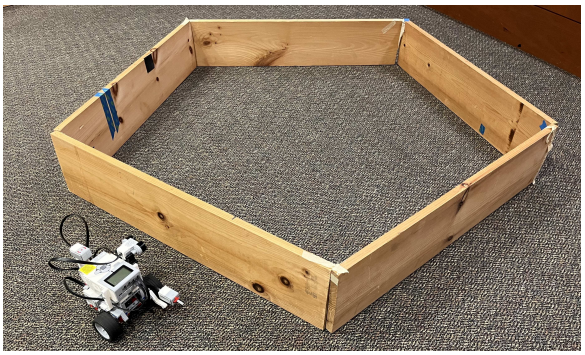


Figure 1: A wheeled Lego robot navigating around an obstacle as part of an undergraduate robotics lab assignment.

To enhance mastery and reduce stress, we incorporated voluntary retakes into our class, allowing students to repeat each lab assignment once. Retakes foster an environment where mistakes are normalized and part of the learning process. This encourages students to practice until they master class content [1]. It also allows students to practice debugging their code—an important yet challenging skill essential for developing robots [3]. Operating a robot in the real world often introduces unforeseen interactions with its environment, making robots prone to errors that are difficult to debug. Therefore, we wanted to foster an environment encouraging students to use these challenges as a positive learning opportunity through our lab retake process. Our goals were (i) to encourage subject mastery and (ii) to reduce student stress.

We are sharing our strategies for incorporating retakes as well as initial results after the first iteration of this new class setup. Students widely adopted the option of retaking labs, which significantly improved their grades, and they agreed that the option of a retake reduced their stress.

2 BACKGROUND AND RELATED WORK

Retakes have been widely implemented in the context of exams or quizzes [4], but to our knowledge, no analysis of retakes in a robotics lab setting has been done so far. Retakes create a learning environment in which students are allowed to make and learn from mistakes, which is important when learning a new topic. Retakes prepare students for many real-life challenges, where performance refinement is allowable and beneficial [1]. However, retaking assessments only results in learning if the students reflect on their

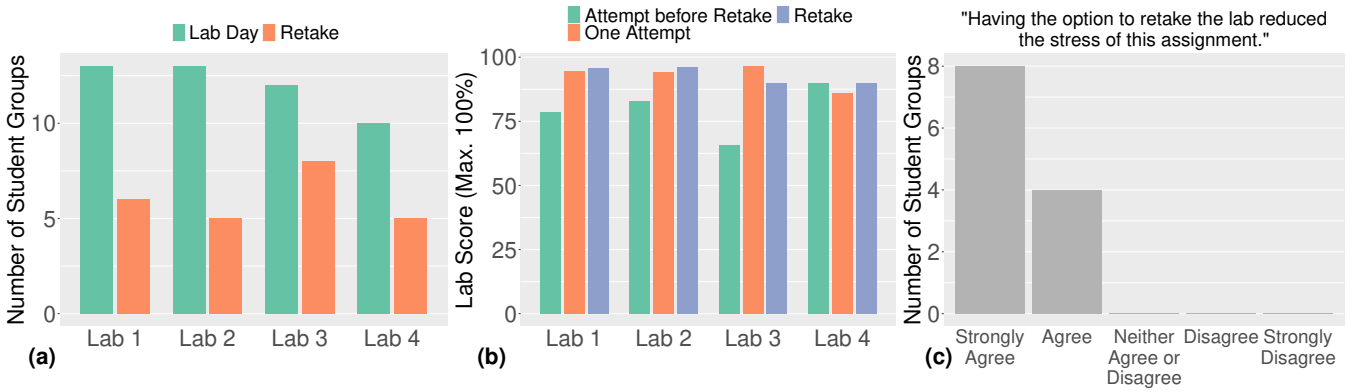


Figure 2: (a) Students utilized lab retakes an average of 1.8 times per group. (b) Groups with lower initial scores chose to retake labs more frequently, scoring similar or better than groups that only performed one attempt. (c) All students agreed that having the option to retake a lab reduced their stress level.

original performance [5]. Furthermore, retakes can require significant effort by instructors, including additional grading, meeting times, and more complex class logistics [1].

3 APPROACH

We implemented the following strategies to incorporate retakes into the class, considering the unique challenges of labs based on physical demonstrations.

Incremental tasks. Lab tasks build on each other, and each previous lab had to be completed to succeed in the next lab. Therefore, students were more motivated to improve their prior labs and to perform well on lab retakes because it would help them in future labs.

Reflection. To motivate students to reflect on their original performance and to analyze potential weaknesses, they were required to fill out a short, ungraded questionnaire explaining what went wrong and how they improved their robots accordingly.

Demonstration. Initial labs occurred during class time, with students watching and discussing their peers' demonstrations. These events were aligned with the suggested schedule for completing the labs to keep students on track. Students were free to skip the original lab and only take the retake to provide flexibility. But this meant they gave up on one additional try and the opportunity to learn from their peers.

Scheduling. We used an online tool to allow students to book individual time slots for retakes. To avoid students forfeiting their chance for a retake because of time constraints, we were very flexible in times offered within a week after the original lab. In addition to the scheduling challenge, setting up a new obstacle course took about 30 min, and we planned 5 min per retake. Even in this relatively small class, it required significant additional time and effort.

Grading. Grades were awarded based on the better score out of the original lab and the retake. There was no risk associated with the retake; grades could only improve to create a learning environment that allows for mistakes while operationalizing new concepts. We shared the grading rubric with the students so they knew which lab evaluation attributed to their final grade.

4 RESULTS

Despite the additional effort, students widely used the retakes. Each student group took at least one retake, and there were, on average, 1.8 retakes per group (standard deviation 0.7), as shown in Fig. 2 (a). Furthermore, there were four instances toward the end of the semester (1 for Lab 3, 3 for Lab 4) where students chose to only participate in the retake but not the original lab. Only two groups participated in both instances for Lab 4.

Across all labs, groups that participated in the retake improved on average by 15.6 out of 100 points. As Fig. 2 (b) shows, groups that chose to participate in the retake had significantly lower average scores in the original labs but reached similar scores or even surpassed the scores of those groups that did not do the retake.

One unforeseen challenge was that some students tailored their retake submission for a particular lab course. They optimized their code and hardware design to work well with the specific obstacle course they had seen during the first round of evaluations, e.g., to stop the robot precisely in the correct location. The robot's location accuracy was worth up to 20% of each lab grade. In the future, we will change the testing environment for the retake.

According to a short survey of the students, all groups 'strongly agreed' or 'agreed' to the statement "Having the option to retake this lab reduced the stress of this assignment" (see Fig. 2 (c)). This is a promising first result in reducing student stress and allowing for mistakes to support content mastery to foster a growth mindset among the students.

REFERENCES

[1] Joe Feldman. 2018. *Grading for equity: What it is, why it matters, and how it can transform schools and classrooms.* Corwin Press.
[2] Bryce Ikeda, Janine Hoelscher, Ron Alterovitz, and Daniel Szafir. 2023. Guiding the Development of Undergraduate Educational Robotics. In *ACM Global Computing Education Conference.* https://doi.org/10.1145/3576882.3617928
[3] Bryce Ikeda and Daniel Szafir. 2021. An AR debugging tool for robotics programmers. In *4th International Workshop on Virtual, Augmented, and Mixed Reality for HRI.*
[4] Laura J Link and Thomas R Guskey. 2022. Is standards-based grading effective? *Theory Into Practice* 61, 4 (2022), 406–417.
[5] Rick Wormeli. 2011. Redos and retakes done right. *Educational Leadership* 69, 3 (2011), 22–26.

Thoughtfully Designing an Industry Mentor Program for Undergraduates in CS and Mathematics

Victoria Hong
St. Joseph's University
Patchogue, New York, USA
vhong@sjny.edu

Cheyne Glass
State University of New York at New Paltz
New Paltz, New York, USA
glassc@newpaltz.edu

ABSTRACT

This work provides a detailed account of a mentoring program pairing junior and senior CS and math undergraduate majors with Industry professionals. Developed with the support of a National Science Foundation S-STEM grant, our Industry Mentor program served high financial needs students, many of whom were first generation college students and/or from groups underrepresented amongst CS or mathematics majors. We present the design of our Industry Mentor program as a whole, as well as our matching process, which was developed in consultation with mentoring experts and applicable literature. The original motivation and objectives for the program, as well as reflections and initial analysis are also shared to provide context and potential claims of impact. Findings contribute to the knowledge base of effective practices for promoting retention, graduation, and career readiness of students in computing and mathematics.

ACM Reference Format:
Victoria Hong and Cheyne Glass. 2024. Thoughtfully Designing an Industry Mentor Program for Undergraduates in CS and Mathematics. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635515

1 INTRODUCTION

Providing students access to an experienced professional in a field aligned with the student's goals, with knowledge of both the written and unwritten rules of life after university, is beneficial to the student's long term professional success [1]. Our Industry Mentor program focused on creating high-impact, low-burden mentoring experiences to foster professional development, career readiness, and job coaching. The mentees included junior and senior undergraduate students with demonstrated financial need pursuing degrees in Computer Science, Computer Information Technology, and Mathematics. While an array of mentoring programs exists within higher education, including faculty-student and peer-to-peer programs, there are very few studies on other types of mentoring relationships [2]. Many of our students were first generation college students with limited access to working professionals in related fields for guidance in career or graduate school preparation. The Industry Mentor program was purposefully designed to help fill the gaps by developing professional relationships for these students.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635515

2 DESIGN AND IMPLEMENTATION

Starting from an established college-readiness mentoring program [3]as a template, we set out to design an Industry Mentoring program for college students which would aim to accomplish the very familiar objectives mentioned above. Along the way, we ensured the program aimed to incorporate certain principles which in our experience are less frequently discussed. A first example is that the students and their industry professional mentors needed time to establish trust and understanding [4] instead of rushing head-on into career-heavy discussions. A second example is that the mentoring program should not cause more harm than good [5]. A network of potential industry mentors was built out, mostly with personal ties to our faculty to maximize the trust we had in them to play such vital roles for our students. During the students' junior year, both they and potential mentors filled out surveys which were used to match them. Not only did we consider professional criteria, but also demographic preferences (such as race or gender) whenever students expressed these as being particularly important to them. We learned that assuming demographic preferences was just as harmful as not considering them at all. Once matched, mentors were oriented in a group meeting, where they were told about program expectations: one hour or more of video-conference meeting(s) a month, where they would try to incorporate certain themes into the discussion as described in a monthly newsletter to mentors. Mentors were asked to complete brief surveys after each meeting. To ensure the commitment from the mentors we only made matches after the mentors confirmed their interest through multiple phases; again trying to avoid our program causing harm. The curriculum of suggested themes (promotion/negotiation, code-switching, work-life balance, resume-building, career navigation, interviewing, imposter syndrome) were arranged in a way where important topics were cycled through according to the students' academic calendar and the stage of relationship building between the pair. For the majority of the first year, the conversations were encouraged to be lighter and more focused on building the relationship, while towards the end of the spring semester of their junior year, the students were encouraged to start engaging in more career-focused thought exercises.

3 REFLECTIONS AND ANALYSIS

While much of our descriptive analysis would require additional comparative data, initial findings suggest that the following statistics indicate improvement from the general student body within

our department. The data comes from 26 student participants across two cohorts from fall 2020 to spring 2023. Of the students participating in the Industry Mentor program, 96% graduated within 5 years compared to our University 6-year graduation rate of 69%. While 23% of our department's students eligible for internships participated in the Industry Mentor program, 60% of all internships were completed by students in the Industry Mentor program. Within one month of graduation, 10 of the 23 students in the Industry Mentor program were employed in professional careers related to their majors, and 5 were enrolled in graduate studies. Of those students from this program who are employed, 30% were offered full time employment upon graduation by their Industry mentors. On a satisfaction scale of 1 (lowest) to 10 (highest) the mentors' satisfaction with the Industry Mentor program rating median was a 9 and the mentees' rating median was also 9. In summative surveys, both mentors and mentees suggested that having at least one meeting in person would benefit the program.

4 FUTURE WORK

In the future we think more care in designing the frequency of the Industry Mentor meetings, as well as increasing our professional network to draw more mentors, and revising the list of topics based on suggestions from the mentor/mentee surveys, would all lead to improved outcomes. Based on mentor/mentee feedback, we would also seek funding for at least one in person Industry mentor/mentee event per academic year. More comprehensive quantitative analysis would be helpful in analyzing which aspects of the design were crucial to positive outcomes.

5 ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1741818.

REFERENCES

[1] G. Crisp and I. Cruz, "Mentoring college students: A critical review of the literature between 1990 and 2007," *Research in higher education*, vol. 50, pp. 525–545, 2009.
[2] M. Dollinger, S. Arkoudis, and S. Marangell, "University alumni mentoring programs: a win-win?" *Journal of Higher Education Policy and Management*, vol. 41, no. 4, pp. 375–389, 2019.
[3] L. Merrill, N. Siman, S. Wulach, and D. Kang, "Bringing together mentoring, technology, and whole-school reform: A first look at the imentor college ready program. report." *Research Alliance for New York City Schools*, 2015.
[4] A. Bayer, J. B. Grossman, and D. L. DuBois, "Using volunteer mentors to improve the academic outcomes of underserved students: The role of relationships," *Journal of Community Psychology*, vol. 43, no. 4, pp. 408–429, 2015.
[5] J. Rhodes, B. Liang, and R. Spencer, "First do no harm: Ethical principles for youth mentoring relationships." *Professional psychology: Research and practice*, vol. 40, no. 5, p. 452, 2009.

Towards Attention-Based Automatic Misconception Identification in Introductory Programming Courses

Muntasir Hoq
North Carolina State University
United States
mhoq@ncsu.edu

Jessica Vandenberg
North Carolina State University
United States
jvanden2@ncsu.edu

Bradford Mott
North Carolina State University
United States
bwmott@ncsu.edu

James Lester
North Carolina State University
United States
lester@ncsu.edu

Narges Norouzi
University of California Berkeley
United States
norouzi@berkeley.edu

Bita Akram
North Carolina State University
United States
bakram@ncsu.edu

ABSTRACT

Identifying misconceptions in student programming solutions is an important step in evaluating their comprehension of fundamental programming concepts. While misconceptions are latent constructs that are hard to evaluate directly from student programs, logical errors can signal their existence in students' understanding. Tracing multiple occurrences of related logical bugs over different problems can provide strong evidence of students' misconceptions. This study presents preliminary results of utilizing an interpretable state-of-the-art Abstract Syntax Tree-based embedding neural network to identify logical mistakes in student code. In this study, we show a proof-of-concept of the errors identified in student programs by classifying correct versus incorrect programs. Our preliminary results show that our framework is able to automatically identify misconceptions without designing and applying a detailed rubric. This approach shows promise for improving the quality of instruction in introductory programming courses by providing educators with a powerful tool that offers personalized feedback while enabling accurate modeling of student misconceptions.

ACM Reference Format:
Muntasir Hoq, Jessica Vandenberg, Bradford Mott, James Lester, Narges Norouzi, and Bita Akram. 2024. Towards Attention-Based Automatic Misconception Identification in Introductory Programming Courses. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635575

1 INTRODUCTION

To effectively support students' learning of introductory programming, it is essential to identify and address their misconceptions of fundamental concepts in a timely manner [7]. However, the disproportionate growth in the number of Computer Science (CS) students compared to teaching staff makes it infeasible to monitor student programs to infer misconceptions in real-time [1]. Automated identification of student misconceptions could be an efficient way to provide individualized feedback either through teachers or automatically. However, automated identification of misconceptions is challenging as misconceptions are primarily latent constructs that cannot be assessed directly. Nevertheless, misconceptions lead to logical errors in student code [2]. Tracing patterns of student errors over time can lead to effectively identifying misconceptions in student programs [4].

Prior studies have attempted semi-automated misconception discovery by highlighting patterns using association rules [3, 5]. However, these studies only focused on simple association rules that fall short in analyzing more complex tasks, i.e., programming [8]. One recent study [8] utilized deep learning models to identify misconceptions automatically. However, their approach required manual labeling of data using an expert-designed rubric. Moreover, their methodology detects misconceptions on a per-rubric-item basis within the student code. This approach does not effectively address misconceptions that may span across multiple rubric items or in cases where rubric items are missing. In this study, we take an initial step towards the automated identification of student misconceptions by showing preliminary results from a deep learning model that can automatically identify multiple logical errors in students' code. In our approach, we build a classifier to group student programs into correct vs. incorrect categories. The dataset is labeled automatically by running programs through a set of test cases. Our framework uses a modified version of a state-of-the-art interpretable deep learning model, Subtree-based Attention Neural Network (SANN) [6]. We use the attention layer of SANN to highlight program snippets in student code that are most influential in classifying code as correct vs. incorrect. A preliminary evaluation of this approach suggests that these patterns are closely associated with logical errors in student code. We demonstrate the outcomes of our framework in this study.

2 DATASET AND METHODOLOGY

We use a publicly-available dataset[1] collected from the CodeWorkout platform. The dataset contains student programming solutions in Java for 50 programming problems from a CS1 course. These submissions were scored in a range of [0,1] based on the number of

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635575

[1] https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=3458

test cases passed. To train our model, we categorized the data into correct or incorrect for a binary classification. In this experiment, we work with one problem named *"caughtSpeeding"* containing 1,574 student programs (correct: 617, incorrect: 957). Uncompilable solutions were excluded from the dataset due to their inability to generate Abstract Syntax Trees (ASTs) and chose not to consider syntactical errors in this study. Using SANN, we map student programs' AST representations to their corresponding condensed vector representation. SANN leverages an attention mechanism that enables us to understand which program subtrees are responsible for the model output, providing interpretability power. While the original SANN model used a genetic search optimization process to break down the AST into non-overlapping subtrees of fixed sizes given a specific task, we modified the original SANN model to incorporate all possible subtrees with various sizes. This decision was made because subtrees of various sizes can represent logical errors, and thus, a fixed-size subtree might not comprehensively encapsulate a single error. Furthermore, logical errors can have a hierarchical nature, where a larger subtree representing a logical error can include another subtree representing a smaller error.

3 RESULTS AND DISCUSSION

In this study, we train the modified SANN model on student program correctness (incorrect: 0, correct: 1). We divide the dataset into train, validation, and test sets (60%, 20%, 20%). We set the embedding size to 64 and kept the other parameters with their default values. The accuracy of the model on the test set is 91%.

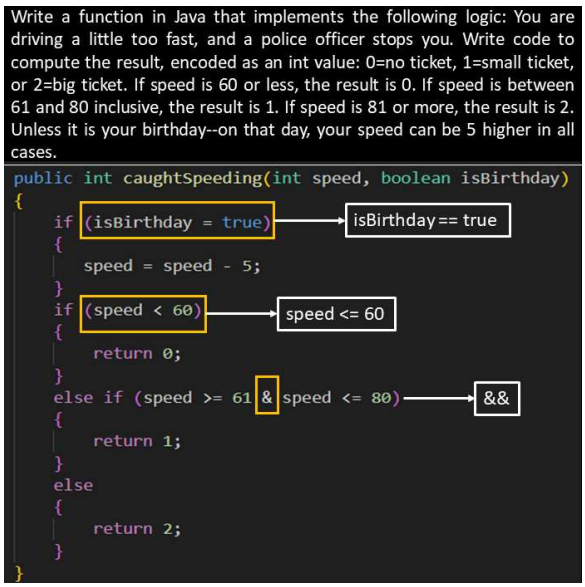


Figure 1: Incorrect student solution for caughtSpeeding.

Using the trained model, we extract attention weights for each possible subtree to highlight the most influential subtrees for the predictions made by the model. We hypothesized that subtrees representing logical errors should receive high weights by the attention mechanism. We did a proof of concept to investigate our hypotheses further. Based on our observations, the attention mechanism correctly identifies erroneous parts of incorrect programs. Figure 1 illustrates one instance of an incorrect student solution with three errors to the *caughtSpeeding* problem. The errors are marked in orange, and their corrected versions are marked in white in the figure. As shown, the model successfully identifies all of the errors in the code. This study is a stepping stone to a generalized programming misconception identifier for CS1. This automated approach has the potential to help enhance instruction quality by enabling educators to pinpoint and address individual student misconceptions, thus promoting a more effective and tailored learning experience.

4 LIMITATIONS AND FUTURE DIRECTIONS

The results of this preliminary study show promise for utilizing our model to highlight logical errors in students' solutions. This can serve as a first step towards automated identification of students' misconceptions and highlight key areas of difficulty when their progress is tracked over time. Utilizing the results of this model, we plan to employ clustering techniques to group similar logical errors to facilitate feedback propagation for instructors. Furthermore, an aggregated report on students' common logical errors and misconceptions could assist educators in identifying and assessing the most challenging concepts, prompting a reevaluation of the corresponding course materials. In the future, we plan to formally evaluate the effectiveness of our model by comparing its results against a set of ground truth logical errors identified by experts for each problem. Our results show our model's effectiveness in identifying logical errors from solutions to a single problem. We plan to evaluate the generalizability of our model by applying it to a dataset that contains solutions to multiple problems. Furthermore, our model can identify the absence of an important programming construct (i.e., a student missing a part of the solution) by highlighting the whole Abstract Syntax Tree for that part. We plan to incorporate information from correct solutions to identify missing constructs in incorrect solutions as well.

5 ACKNOWLEDGMENTS

This research was supported by the National Science Foundation (NSF) under Grants DUE-2236195 and DUE-2331965.

REFERENCES

[1] Bita Akram, Wookhe Min, Eric Wiebe, Anam Navied, Bradford Mott, Kristy Elizabeth Boyer, and James Lester. 2020. Automated assessment of computer science competencies from student programs with gaussian process regression. In *EDM'20*.
[2] Ella Albrecht and Jens Grabowski. 2020. Sometimes It's Just Sloppiness-Studying Students' Programming Errors and Misconceptions. In *SIGCSE'20*. 340–345.
[3] Myse Elmadani, Moffat Mathews, and Antonija Mitrovic. 2012. Data-driven misconception discovery in constraint-based intelligent tutoring systems. (2012).
[4] Andrew Ettles, Andrew Luxton-Reilly, and Paul Denny. 2018. Common logic errors made by novice programmers. In *ACE'18*. 83–89.
[5] Eduardo Guzmán, Ricardo Conejo, and Jaime Gálvez. 2010. A data-driven technique for misconception elicitation. In *UMAP'10*. 243–254.
[6] Muntasir Hoq, Sushanth Reddy Chilla, Melika Ahmadi Ranjbar, Peter Brusilovsky, and Bita Akram. 2023. SANN: Programming Code Representation Using Attention Neural Network with Optimized Subtree Extraction. In *CIKM'23*. 783–792.
[7] Samiha Marwan, Bita Akram, Tiffany Barnes, and Thomas W Price. 2022. Adaptive immediate feedback for block-based programming: Design and evaluation. *IEEE Transactions on Learning Technologies* 15, 3 (2022), 406–420.
[8] Yang Shi, Krupal Shah, Wengran Wang, Samiha Marwan, Poorvaja Penmetsa, and Thomas Price. 2021. Toward semi-automatic misconception discovery using code embeddings. In *LAK'21*. 606–612.

Analyzing Student Performance with Free Late Submission Days

John R. Hott
jrhott@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

ABSTRACT

We investigate the effects of a flexible late policy on student performance submission behavior across two semesters of a lower-level required course (LL) and an upper-level elective (UL). The first semester late policies in both courses were strict: LL incorporated a 10% penalty per day for two days; UL rarely allowed late submissions. In each course, the late policy was relaxed in the second semester to provide two no-penalty late days for each assignment.

Our results show that in the LL class with the strict late policy, grades decrease each day past the deadline. When given two no-penalty late days, students in LL tended to submit their assignments later, with 41% of submissions coming after the deadline. While students' homework scores were higher than their peers with the stricter policy (before penalties) on the first day after the deadline, their scores on the second day of submission were equivalent. With more late submissions, the overall homework scores in LL were lower with the no-penalty late days (controlling for late penalties). In contrast, students in UL submitting two days late in the second semester performed similarly to their peers in the first semester with the stricter penalty, while average scores for assignments submitted on (and around) the deadline improved dramatically. These results suggest that providing an extra day with no penalty is helpful for students, such as those making just-in-time submissions, but that additional days may lower performance overall for students in lower-level courses.

ACM Reference Format:
John R. Hott. 2024. Analyzing Student Performance with Free Late Submission Days. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635562

1 INTRODUCTION

Students procrastinate [4], causing added stress as deadlines approach. We employed a flexible late policy that provides students the option to submit assignments up to two days late without a penalty. We examined student submission patterns and how students perform under this more flexible policy compared with stricter policies such as a daily late penalty on grades or disallowing any late submissions without instructor approval. Specifically, we investigated: (RQ1) Does allowing late submissions without penalty improve student performance on assignments? (RQ2) When given time after the deadline to submit, do student submit later? How does this compare between upper- and lower-level students?

2 RELATED WORK

Shaffer and Edwards [5] found that students who scored higher on programming projects tended to start earlier, and finish earlier, than those who scored lower. Many approaches, such as gamification [3] and mastery learning [2], either provide incentives to submit early or provide means to submit late to improve student outcomes. Tyler et al [6] investigated the effects of providing 3 late days per semester and extended deadlines for early submitters, finding that students who started before the early deadline earned the extra day and performed better than those who did not. After conducting a student and faculty survey, Santelli et al [4], concluded that both groups found an institutional policy of 10% late penalty per day to be both effective and fair, but noted that perceptions of the flexibility of the policy differed between groups.

3 METHODOLOGY

We collected data[1] from two courses at the University of Virginia spanning two semesters each. One course was an upper-level elective (UL) in the fall 2021 and spring 2022 semesters; the other was a lower-level foundation course (LL) in the fall 2022 and spring 2023 semesters. We define (i) a *strict* late policy as one that either does not allow late submissions or imposes a penalty after the deadline and (ii) a *flexible* late policy as consisting of two additional days after the deadline in which students can submit assignments without any grade penalty. For the flexible policy, the instructor informed students of the policy when discussing the syllabus and occasionally throughout the semester, but only provided the official deadlines in the course schedule and when reminding students about the homework assignments.

UL, a project-based elective course covering programming languages for web applications, included 7 homework assignments which accounted for 20-30% of the overall course grade each semester. In the fall semester (84 students), the course followed a *strict* policy that did not allow late submission, except for extenuating circumstances. In the spring semester (114 students), the course used the *flexible* policy. In both semesters, one homework grade was dropped.

LL is a second-year required course for the Computer Science major at the University of Virginia that covers binary representation, computer organization and architecture, assembly programming, and an introduction to C. Students in LL were asked to independently submit 10 homework assignments, accounting for 40% of the overall course grade. We included 7 LL homework assignments that were comparable across semesters. In the fall semester (430

[1]This study was approved under the University of Virginia Institutional Review Board (IRB)-SBS protocol number 5958.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635562

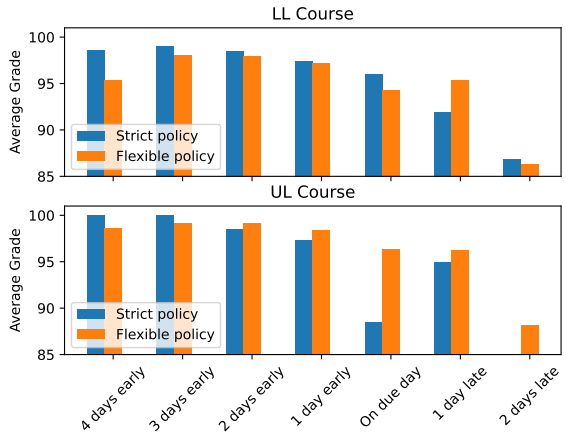


Figure 1: Average scores for final submissions on each day relative to the deadline for LL (top) and UL (bottom).

students), the course followed a *strict* penalty-based late submission policy with a 10% penalty per day for two days. The spring semester (480 students) followed the same *flexible* policy used in UL's second semester.

3.1 Analysis

We collected the timestamp of the final submission and the final grade received for each student on each homework. We identified and aligned the deadlines for each assignment to group final submissions based on when they occurred relative to the deadline, up to four days before and two days after each assignment was due. All raw scores were converted into grade percentages, using the pre-penalty raw scores under the stricter policies.

To address our research questions, we calculated the average score for all assignments on each day relative to the deadline (RQ1, RQ2) and the average number of final submissions per day (RQ2). For each comparison, we performed a two-sample Kolmogorov–Smirnov test [1] to determine significance.

4 RESULTS AND DISCUSSION

The average scores for each day (RQ1) are depicted in Figure 1. A statistically significant increase of 3.42% is evidenced in scores one day after the deadline with the flexible policy in LL. However, there was a slight decrease in scores for submissions on the due date and two days late. Average scores in UL increased under the flexible policy for two days before and one after the due date, with a significant increase of 7.87% on the due date. However, students in both classes performed lower overall in most (UL) or all (LL) assignments under the flexible policy than under the stricter policy.

Students made their final submissions later in both classes under the flexible policy (RQ2). Over 32% of LL submissions and 27% of UL submissions were made two days late. In LL, the average score on days surrounding the deadline follows a similar pattern: a general downward trend starting 2-3 days before the deadline and continuing through two days late. Notably, even though LL students with flexibility performed better one day late, the average scores for students submitting two days late are comparable under both policies—86.87% vs 86.31%. Since the bulk of submissions shifted from on time (38.76% of all submissions under the strict policy) to two days late (32.21% of all submissions with flexibility), therefore overall scores in LL tended to be lower with the more flexible late policy. In contrast, students in UL submitting two days late with the flexible policy scored most similarly to those submitting on the due date under the strict policy—88.18% vs 88.45% respectively—with a similar trend observed on prior days. These results indicate that flexibility is needed, as scores surrounding the due dates increased, but should include limitations or encouragement for students to not procrastinate until the final submission deadline. Additionally, the difference in score pattern between LL and UL suggests that students later in their studies may be better equipped to handle flexible late policies than students in their first or second year.

5 CONCLUSION AND FUTURE WORK

We examined student performance across strict and flexible late submission policies for both early-career and late-career college students. We found that when given the option to submit up to two days late without penalty, many students tended to submit later—after the deadline. For students in their first or second year of college, their performance decreased when given this flexibility. In contrast, students in their third or fourth year who submitted late performed similarly to on-time submissions under stricter policies. We also noted that both courses evidenced a significant increase in scores submitted on or just after the deadline when given the option to submit late. Further work is indicated to determine if those submissions consist of students who needed slightly more time to make their final submission—i.e., those who tried to submit by the deadline but found errors requiring re-submission a little late—or students who used more of the late day to improve their work—i.e., those who needed additional time due to assignments in other courses. Regardless, due to this slight increase, our results suggest the use of limited flexibility rather than a more strict policy overall may improve student outcomes.

REFERENCES

[1] 2008. Kolmogorov–Smirnov Test. In *The Concise Encyclopedia of Statistics*. Springer New York, New York, NY, 283–287. https://doi.org/10.1007/978-0-387-32833-1_214

[2] James H Block and Robert B Burns. 1976. Mastery learning. *Review of research in education* 4 (1976), 3–49.

[3] Stephen H. Edwards and Zhiyi Li. 2020. A Proposal to Use Gamification Systematically to Nudge Students Toward Productive Behaviors. In *Proceedings of the 20th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '20)*. Association for Computing Machinery, New York, NY, USA, Article 28, 8 pages. https://doi.org/10.1145/3428029.3428057

[4] Beverly Santelli, Sarah N Robertson, Elizabeth K Larson, and Samia Humphrey. 2020. Procrastination and Delayed Assignment Submissions: Student and Faculty Perceptions of Late Point Policy and Grace within an Online Learning Environment. *Online Learning* 24, 3 (2020), 35–49.

[5] Clifford A. Shaffer and Stephen H. Edwards. 2011. Scheduling and Student Performance. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education* (Darmstadt, Germany) *(ITiCSE '11)*. Association for Computing Machinery, New York, NY, USA, 331. https://doi.org/10.1145/1999747.1999842

[6] Jeramey Tyler, Matthew Peveler, and Barbara Cutler. 2017. A Flexible Late Day Policy Reduces Stress and Improves Learning (Abstract Only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 718. https://doi.org/10.1145/3017680.3022439

Towards More Efficient Office Hours for Large Courses: Using Cosine Similarity to Efficiently Construct Student Help Groups

John R. Hott*
jrhott@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

Mark Floryan
mfloryan@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

Nada Basit
basit@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

ABSTRACT

As undergraduate enrollment in computer science rises, instructors continue to investigate methods to improve the student experience at scale. One aspect commonly used in courses at scale is queue-driven office hours, in which students join an online queue and meet with teaching assistants on a first-come, first-serve basis (FIFO).

This poster introduces a novel office hours queue feature that automatically groups students in office hours using the cosine similarity metric across their reported issues provided upon joining the queue. Using real office hour attendance data from a 480-person undergraduate course as a basis for simulation, we find that moderate decreases in student wait time during the semester overall (11% on average) are possible, with more significant decreases possible on the busiest days (20% on average). This approach is suitable for real-world testing and these gains are possible without asking students to provide any additional information than they already do when attending office hours. Therefore, this work provides motivation for implementation of such an approach in future courses.

ACM Reference Format:
John R. Hott, Mark Floryan, and Nada Basit. 2024. Towards More Efficient Office Hours for Large Courses: Using Cosine Similarity to Efficiently Construct Student Help Groups. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635544

1 INTRODUCTION

According to the 2022 Taulbee Survey [1], undergraduate enrollment in Computer Science grew 10.9% from 2021. The growth of programs in Computer Science has steadily increased over the last decade and institutions continue to adapt to larger enrollments. Large courses often employ office hours with queuing systems (e.g., My Digital Hand [3]); there is interest in both predicting the wait times of students during these help sessions [4] and improving the efficiency of teaching assistants (TAs) in helping students.

One mechanism to decrease wait times is grouping students with similar issues together with a TA. This work investigates the feasibility of using algorithms—specifically the cosine similarity measure [2]—to intelligently group students with similar problems in a live office hours scenario. Grouping must be computed in a low-data setting, i.e., students do not add any additional information than would normally be expected.

2 TOOLS AND METHODS

During the spring 2023 semester[1], students in a 480-person *Computer Systems and Organization* course used an office hours queue tool shown in Figure 1, which recorded the students' primary actions in the system. This included the time the student (i) joined the queue, (ii) was helped by a TA, and (iii) had their meeting ended by the TA. Students joining the queue were required to enter a subject and description of their issue. The tool then connected one student in the queue with one TA: first-come, first-serve. There were 3,945 help sessions recorded during the semester.

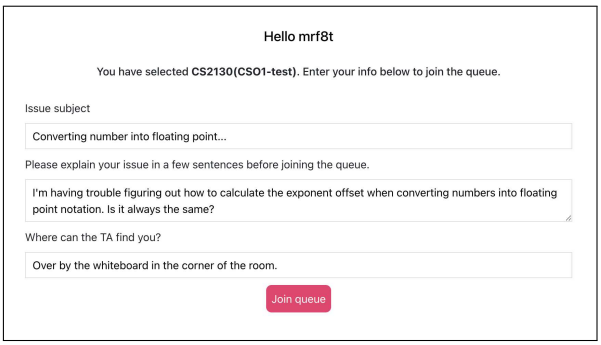


Figure 1: The interface students used when entering the office hours queue. Students must enter a subject, a description of the issue they are having, and their physical location.

2.1 Measuring the Similarity of Student Issues

In order to explore the potential time benefit of grouping students in office hours, a method for detecting similar issues was needed. **Cosine similarity** [2], a widely-used measure that computes the similarity between two vectors in an inner-product space, provides such a method. It can then be used to detect the similarity between sentences in a corpus of text via a two step process. In step one, Term Frequency-Inverse Document Frequency (TF-IDF) vectorizes text documents (here, student issues), converting student-reported issues into input vectors, while saturating out common words. TF-IDF is also robust against the variable lengths of student-reported

*All authors contributed equally to this research.

[1]This study was approved under the University of Virginia Institutional Review Board (IRB)-SBS protocol number 5685.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635544

Table 1: The parameters of the simulation. Default values below simulate office hours with no grouping and approximate the results that were seen in reality.

Variable	*Abbr.*	*Description*	*Default*	*Simulation Values*
Match Threshold	MT	Cosine similarity must return a match of at least this value in order to consider the two student issues matched	1.0	0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
Grouped Probability	GP	When a TA visually inspects the match, this is the probability the TA will confirm the group looks feasible, otherwise the TA rejects the member from the group	0.0	0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
Max Group Size	MGS	This is the maximum number of group members allowed during the simulation	1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Group Duration Multiplier	GDM	Multiplier that is incorporated into the time it takes to help a group	1.0	1.0, 1.2, 1.4, 1.5, 1.6, 1.8, 2.0

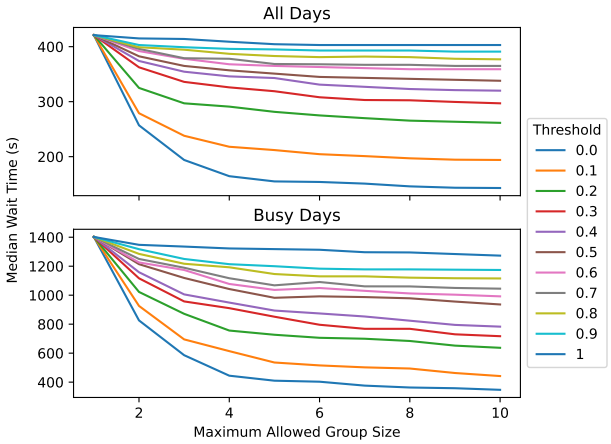


Figure 2: Median wait times across all days (top) and busy days (bottom) as similarity threshold increases across allowed group sizes.

issues. In step two, the angle between these vectors can be computed to determine the similarity between issues.

2.2 Efficacy of Cosine Similarity Grouping

Office hour help interactions between students and TAs were **simulated** using cosine similarity grouping based on actual timestamps of TA arrival and departure times, student arrivals, time spent with a TA, student reported issues and more, as observed in the **real-world data**. By simulating over actual recorded events, our results depict what may happen in a similar semester if students were grouped by cosine similarity instead and determine the overall impact on wait times. The observed data also provides ground truth for the simulation. Several parameters, as seen in Table 1, allowed the simulation to incorporate different degrees of student grouping. When executed with *default* parameters shown in the table, the simulation approximates the actual behavior seen in office hours, with each TA helping exactly one student, in order of their arrival.

3 RESULTS

Figure 2 shows the change in median wait time when two variables of interest (*match threshold* and *max group size*) are altered over the semester. The upper left corner of the plots denote the baseline median wait time without grouping: ~7 minutes across all days and ~23 minutes on busy days. *Busy days* are defined as *any date in which the maximum queue size at any point during the day exceeded the average maximum daily queue size for the semester (10 students).* As the allowable *max group size* increases, more students are able to be placed in groups, and overall wait times decrease.

Since low cosine similarity thresholds are unlikely to be useful in practice, due to matching students with unrelated issues, a threshold of 0.6 (60%) will provide more accurate groupings. Additionally, once group sizes exceed 5 students, the potential savings decrease significantly. For these reasons, cosine similarity threshold of 0.6 and max group size of 5 appear to provide the best balance between increasing efficiency in office hours while ensuring group sessions are likely to be productive. Median wait time savings ranges from ~1 minute (all days) to ~6 minutes on the busiest days. Therefore, the students who waited the longest stand to benefit the most from this approach.

4 DISCUSSION AND CONCLUSION

Overall, the results show that using cosine similarity to assist TAs in grouping students during office hours is likely to lead to a better overall office hours experience, as simulated on real data from students in a past semester. Median wait time should drop by between 1 and 6 minutes per student, and more on busy days. Students can also expect to spend slightly more time being helped by a TA (albeit in a group setting), which might mitigate some of the lack of personalized help a group setting affords. There is concern that TAs may spend significantly more time with groups, but simulation suggests that student attendance patterns are resistant to this increase. These results are promising enough to move forward testing this approach in a real classroom, which will occur in fall 2023.

REFERENCES

[1] Computing Research Association (CRA). 2022. The Taulbee Survey. *https://cra.org/crn/wp-content/uploads/sites/7/2023/05/2022-Taulbee-Survey-Final.pdf* (2022), Aug. 6, 2018.
[2] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. 2012. Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, Vol. 4. 1.
[3] Aaron J Smith, Kristy Elizabeth Boyer, Jeffrey Forbes, Sarah Heckman, and Ketan Mayer-Patel. 2017. My digital hand: A tool for scaling up one-to-one peer teaching in support of computer science learning. (2017), 549–554.
[4] Nick Troccoli, Brahm Capoor, and Michael Troute. 2017. Predicting CS106 Office Hours Queueing Times. *Past CS229 Project* (2017).

Integrating Personalized Parsons Problems with Multi-Level Textual Explanations to Scaffold Code Writing

Xinying Hou
University of Michigan
Ann Arbor, Michigan, USA
xyhou@umich.edu

Barbara J. Ericson
University of Michigan
Ann Arbor, Michigan, USA
barbarer@umich.edu

Xu Wang
University of Michigan
Ann Arbor, Michigan, USA
xwanghci@umich.edu

ABSTRACT

Novice programmers need to write basic code as part of the learning process, but they often face difficulties. To assist struggling students, we recently implemented personalized Parsons problems, which are code puzzles where students arrange blocks of code to solve them, as pop-up scaffolding. Students found them to be more engaging and preferred them for learning, instead of simply receiving the correct answer, such as the response they might get from generative AI tools like ChatGPT. However, a drawback of using Parsons problems as scaffolding is that students may be able to put the code blocks in the correct order without fully understanding the rationale of the correct solution. As a result, the learning benefits of scaffolding are compromised. Can we improve the understanding of personalized Parsons scaffolding by providing textual code explanations? In this poster, we propose a design that incorporates multiple levels of textual explanations for the Parsons problems. This design will be used for future technical evaluations and classroom experiments. These experiments will explore the effectiveness of adding textual explanations to Parsons problems to improve instructional benefits.

CCS CONCEPTS

• **Human-centered computing**; • **Applied computing** → **Interactive learning environments**;

KEYWORDS

Introductory Programming, Code Explanations, Parsons Problems, Code Writing, Scaffolding, Hint, Large Language Models

ACM Reference Format:
Xinying Hou, Barbara J. Ericson, and Xu Wang. 2024. Integrating Personalized Parsons Problems with Multi-Level Textual Explanations to Scaffold Code Writing. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635606

1 INTRODUCTION

One main goal of introductory programming courses is to develop basic coding skills. Although AI generation tools are now widely used in professional code development, it is still crucial for beginners to participate in code-writing activities to fully acquire fundamental programming concepts. In recent work, we explored providing Parsons problems as scaffolding for students who are struggling while writing code independently in Python [2–4]. By leveraging the power of LLMs, we designed and implemented a system called *CodeTailor* to provide multi-stage personalization through a Parsons problem to assist students with code writing (Figure 1). An initial study with 18 novice programming students showed that CodeTailor is engaging and benefits learning. However, some students reported challenges as they could not fully understand the meaning of some code blocks. Textual explanations could help users understand the program's components, objectives, and structure [5]. They can assist beginners in learning more from the Parsons problem, providing an additional opportunity to read code blocks, understand the problem, and comprehend the solution. Hence, we propose a design to incorporate multiple levels of natural language explanations into CodeTailor. While explanations have been given with various programming learning materials or activities, there is limited research on integrating them with Parsons problems. Given the increasing interest and demonstrated effects of using Parsons problems to scaffold novice programmers' learning, more research is required to explore the integration of explanations into Parsons problems.

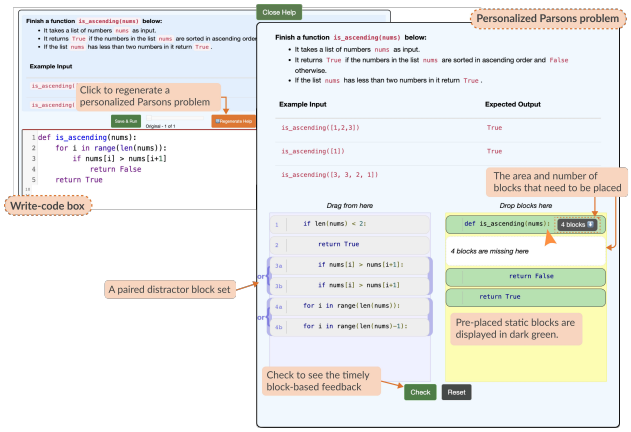


Figure 1: A write-code box (left) with a pop-up personalized Parsons problem as scaffolding (right).

2 SYSTEM DESIGN

CodeTailor is a large language model (LLM)-powered system that provides real-time, on-demand, and multi-staged personalized Parsons problems to support students while writing code. It differs from existing hint systems for programming in that CodeTailor provides an active learning opportunity through the help. Specifically, students need to engage in problem-solving with the support

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635606

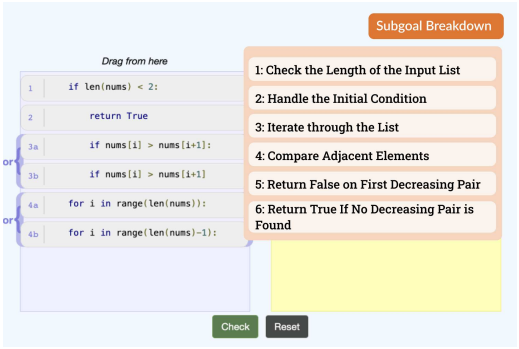


Figure 2: A subgoal list about the Parsons problem solution.

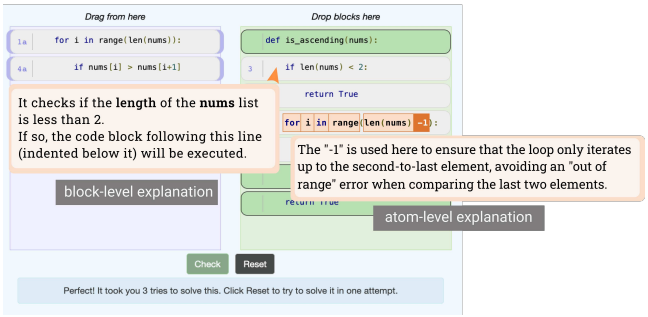


Figure 3: Block-level and atom-level explanations for the finished Parsons blocks.

provided rather than merely being consumers of a displayed help message [1]. An overview of CodeTailor's main interface is shown in Figure 1. After clicking the "Help" button, CodeTailor provides a personalized Parsons problem that is dynamically generated based on the student's latest code state. The personalized Parsons problem is partially complete, with the correct student-written code lines already in place and unmovable. The distractors, which are unnecessary blocks in the correct solution, are generated from the student's own incorrect code lines. Furthermore, CodeTailor adjusts the difficulty of the Parsons problem based on students' requests. If a student fails three times with a fully movable Parsons problem, CodeTailor allows them to combine two code blocks into one. Once the student successfully solves the Parsons problem, they can copy the solution to the write-code box.

We enhance CodeTailor by incorporating multiple levels of LLM-generated textual explanations. Four types of explanations will be generated and inserted in real-time, enabling CodeTailor to provide personalized explanations accompanying the Parsons problem. We are still exploring methods to assess the quality of these explanations automatically.

Subgoal guidance explanation: Students will have access to a subgoal summary of the solution provided in the unsolved Parsons problem. This is presented in numbered bullet points, breaking down the current programming task into 4-6 more manageable task subgoals (Figure 2). The subgoal guidance aims to help students gain a high-level abstract understanding and guide those who struggle to start solving a Parsons problem.

Block-level code explanation: After solving the Parsons problem, students can hover over each block to get an explanation of each block. These explanations clarify the behavior and purpose of the Parsons solution at the block level (Figure 3). For paired distractors, the explanation also provides reasoning for why the correct block is right and why the corresponding distractor block is incorrect.

Atom-level code explanation: For each block in the Parsons solution, students can also click on an atom (individual elements of the programming language, such as keywords and statements) within a code block to receive an explanation of the atom's text surface, execution (if any), and purpose (Figure 3).

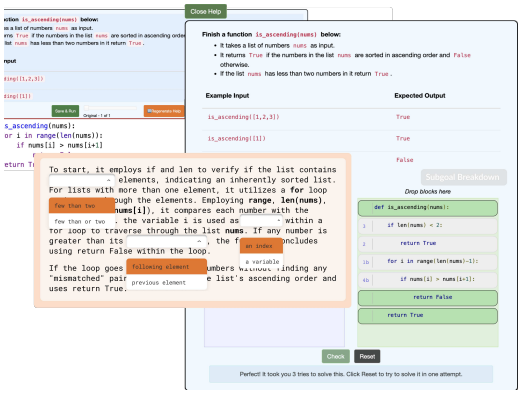


Figure 4: Students will receive a menu-based self-explanation question after solving the Parsons problem.

A self-explanation question to reflect on the Parsons solution: After getting the write-code problem correct, this Parsons solution will appear again without any textual explanations. This time, students need to explain the reasoning behind this solution in a menu-based self-explanation prompt. The prompt is designed to minimize guessing and enhance students' understanding. In this case, an LLM will generate the main structure of this explanation, leaving keywords to be filled in (Figure 4) by the student.

ACKNOWLEDGMENTS

The funding for this research came from the National Science Foundation award 2143028. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Michelene TH Chi and Ruth Wylie. 2014. The ICAP framework: Linking cognitive engagement to active learning outcomes. *Educational psychologist* 49, 4 (2014), 219–243.

[2] Xinying Hou, Barbara Jane Ericson, and Xu Wang. 2022. Using adaptive parsons problems to scaffold write-code problems. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*. 15–26.

[3] Xinying Hou, Barbara Jane Ericson, and Xu Wang. 2023. Parsons Problems to Scaffold Code Writing: Impact on Performance and Problem-Solving Efficiency. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 2*. 665–665.

[4] Xinying Hou, Barbara J Ericson, and Xu Wang. 2023. Understanding the Effects of Using Parsons Problems to Scaffold Code Writing for Students with Varying CS Self-Efficacy Levels. *arXiv preprint arXiv:2311.18115* (2023).

[5] Cruz Izu, Carsten Schulte, Ashish Aggarwal, Quintin Cutts, Rodrigo Duran, Mirela Gutica, Birte Heinemann, Eileen Kraemer, Violetta Lonati, Claudio Mirolo, et al. 2019. Fostering program comprehension in novice programmers-learning activities and learning trajectories. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*. 27–52.

Building a Mixed-format Computer Science Assessment for Middle School

Jennifer Houchins
jhouchi@wested.org
WestEd
San Francisco, CA, USA

Kim Luttgen
kluttge@wested.org
WestEd
San Diego, CA, USA

Rosalind Owen
rowen@wested.org
WestEd
San Francisco, CA, USA

Lydia Martinez Rivera
lmartin5@wested.org
WestEd
San Antonio, TX, USA

Matt Silberglitt
msilber@wested.org
WestEd
San Francisco, CA, USA

Yvonne Kao
ykao@wested.org
WestEd
San Francisco, CA, USA

ABSTRACT

Despite widespread adoption of the K-12 Computer Science Standards published by the Computer Science Teachers Association (CSTA) in 2017, there remain few validated assessments for computer science that researchers and educators can use to measure students' conceptual understanding. Traditional assessments that do exist tend to consist largely of multiple-choice questions that do not adequately measure students' programming skills or their use of the computer science practices described in the standards. This project aims to develop and iteratively refine a mixed-format assessment for middle school computer science. The first phase of this work unpacked the middle school (6-8 grade band) CS standards to develop prototype items, including live coding tasks, that would assess the learning outcomes addressed therein. In phase 2, we tested the prototype items with students through cognitive interviews to better understand their response processes and the complex problem-solving and reasoning they demonstrated while taking the assessment.

ACM Reference Format:
Jennifer Houchins, Kim Luttgen, Rosalind Owen, Lydia Martinez Rivera, Matt Silberglitt, and Yvonne Kao. 2024. Building a Mixed-format Computer Science Assessment for Middle School. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635578

1 BACKGROUND & RELATED WORK

There are a handful of validated assessments for K-12 computational thinking and computer science, such as the Middle Grades Computer Science Concepts Inventory, Commutative Assessment, the Assessment of Computing for Elementary, or Advanced Placement (AP) Exams [1–4]. All of these, with the exception of the AP CS Exams, are traditional multiple-choice assessments, a format that limits the ability to assess the programming skills and computational practices students have developed. A potential solution to this challenge is to develop a mixed-format CS assessment that combines multiple approaches to identify both students' conceptual knowledge and their programming skills.

2 ASSESSMENT DESIGN

The project has conducted two iterations of mixed-format assessment design. The prototype assessment items consisted of three formats: multiple-choice, constructed response, and live coding tasks. The items are designed to assess students' conceptual knowledge and programming skills related to the following CSTA Standards:

- Use flowcharts and/or pseudocode to address complex problems as algorithms (2-AP-10).
- Create clearly named variables that represent different data types and perform operations on their values (2-AP-11).
- Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals (2-AP-12).
- Systematically test and refine programs using a range of test cases (2-AP-17).

The mixed-format assessment was developed for delivery via Survey Monkey and uses the Scratch platform for the live coding items. Live coding items ask students to create functionality by adding to or modifying an existing program. To capture responses to these items, students logged into generic Scratch accounts created by the research team and remixed the existing program to develop their solutions. Students shared their completed solutions by publishing them, copying the share link, and pasting it into the assessment item within Survey Monkey.

3 COGNITIVE INTERVIEWS

To pilot the assessment, two rounds of cognitive interviews were conducted. The first round of cognitive interviews was conducted with expert programmers to identify any issues with the design of the items and their delivery prior to engaging students. Once identified issues such as spelling or formatting errors were corrected, we purposely recruited middle grades students with some experience with Scratch programming (e.g., at least one computer science course or after-school program that taught Scratch) for a second round of cognitive interviews.

Students' cognitive interviews were conducted virtually via Zoom and video-recorded to assist researchers' note-taking. Students were asked to share their screen while taking the assessment

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635578

and invited to tell researchers about their approaches in answering each assessment item as they completed the assessment. As the students worked, researchers noted items where students appeared to struggle or mentioned aloud having trouble understanding the prompt, answer choices, or concepts. The interviews were limited to one-hour sessions that included time set aside for students to share their thoughts via a set of wrap-up questions. The assessment portion of the cognitive interviews was designed to take approximately 45 minutes, the typical length of one middle school class period.

4 FINDINGS

Results of cognitive interviews with students revealed potential issues with item format, independence, specificity, and complexity. Item format issues took one of two forms and were identified primarily for multiple-choice items. The first format issue was caused by the amount of visual space in items that presented incomplete code, along with options that varied in visual size; students could choose a response based on matching the visual size of the incomplete code and the answer option rather than using their understanding of algorithms and programming. The other was that some items were better suited to a different format entirely. For example, one item that asked students to order a set of steps by entering the corresponding letter of the step (e.g., A, B, C, or D) could have benefited from a drag-and-drop format to facilitate students checking their answer choices before moving to the next item.

One overarching issue, item independence, arose from the development team's attempt to contextualize the items within each task. Some of the context was presented within the items themselves, and subsequent items required students to use this contextual information. In some cases, this way of presenting context caused responses to earlier items to affect the way students responded to later items. The effect was magnified in constructed response items; for example, when students used responses to earlier, multiple-choice items to generate their responses.

Constructed responses also proved challenging for students when the item failed to provide enough specificity about how they were expected to respond. For example, one item asked students to describe what one needs to do to add certain behavior to a program. Students' responses were split between those who responded in paragraph form and those that chose to provide pseudocode. This suggested updates to both the prompt and the scoring rubric for this item.

Finally, issues arising from cognitive demand appeared most often in the live coding items. For example, one live coding item asked students to incorporate behavior dependent on the way a sprite was moving on the screen. This example illustrated that the coordinate system used in coding created a barrier to students demonstrating their understanding of the code concepts. Despite incorporating a compass and axes markings on the program's stage (i.e., where the code behavior is shown), students struggled with the directions in which they were asked to make the sprites move.

5 CONCLUSION & FUTURE WORK

Students' cognitive interviews revealed that mixed-format assessments can be administered with students within the amount of time for a typical class period. Despite the issues described in this work, students found the mixed-format assessment to be engaging in terms of both the context and the format. Students shared that they enjoyed that they were asked to provide code solutions in addition to the multiple-choice and constructed response questions. One student even asked to continue working on their code solution after the conclusion of their cognitive interview.

Aside from students' engagement in the assessment items, evidence of initial content validity was also shown for the assessment. All but two of the multiple-choice items were correctly answered by all the students. The open response item that asked students to describe what was needed to add behavior was also answered correctly by all students, despite the differences in how they provided a response. Finally, the students who participated appeared to understand what they were being asked to do in the live coding items, but struggled to carry out that behavior in the Scratch environment. This suggests that these items do have some evidence of initial content validity, but more study is warranted.

Future work on this type of mixed-format assessment includes another round of item refinement, which is already underway. This round of refinements will address the issues presented here. The next version of the mixed-format assessment will undergo another round of cognitive interviews with students as well as classroom pilots that are planned for the fall and spring of the current academic year (2023-2024).

ACKNOWLEDGMENTS

This work was funded by WestEd.

REFERENCES

[1] AP Central | College Board. 2023. *AP Computer Science Principles Course.* College Board. https://apcentral.collegeboard.org/courses/ap-computer-science-principles/course

[2] Miranda C. Parker, Yvonne S. Kao, Dana Saito-Stehberger, Diana Franklin, Susan Krause, Debra Richardson, and Mark Warschauer. 2021. Development and Preliminary Validation of the Assessment of Computing for Elementary Students (ACES). In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21).* Association for Computing Machinery, New York, NY, USA, 10–16. https://doi.org/10.1145/3408877.3432376

[3] Arif Rachmatullah, Bita Akram, Danielle Boulden, Bradford Mott, Kristy Boyer, James Lester, and Eric Wiebe. 2020. Development and validation of the middle grades computer science concept inventory (MG-CSCI) assessment. *Eurasia Journal of Mathematics, Science and Technology Education* 16, 5 (2020), em1841. https://doi.org/10.29333/ejmste/116600

[4] David Weintrop and Uri Wilensky. 2015. Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs. In *Proceedings of the eleventh annual International Conference on International Computing Education Research* (2015-08-09) *(ICER '15).* Association for Computing Machinery, New York, NY, USA, 101–110. https://doi.org/10.1145/2787622.2787721

Programming Language Knowledge Transfer that Teachers Observe in their Classrooms

Jennifer Houchins
jhouchi@wested.org
WestEd
San Francisco, CA, USA

Rosalind Owen
rowen@wested.org
WestEd
San Francisco, CA, USA

Bryan Matlen
bmatlen@wested.org
WestEd
WestEd, CA, USA

Yvonne Kao
ykao@wested.org
WestEd
San Francisco, CA, USA

ABSTRACT

There has been significant progress in increasing the access to computing education for many K-12 students, including states adopting computer science (CS) standards and/or requiring CS courses. This includes the creation of block-based programming languages to make programming more accessible to younger students. Despite this progress, a new challenge has emerged: Students often struggle to transfer conceptual knowledge when transitioning to a new programming language (e.g., transitioning to a text-based programming after learning a block-based programming language). This poster presents the results of teacher interviews regarding the examples of knowledge transfer they observe in their classrooms. These interviews are part of an overarching project that aims to address the challenge of knowledge transfer between programming languages by developing a framework to support such transfer and deliver curricular supports that can be used to aid students' productive knowledge transfer between programming languages.

ACM Reference Format:
Jennifer Houchins, Rosalind Owen, Bryan Matlen, and Yvonne Kao. 2024. Programming Language Knowledge Transfer that Teachers Observe in their Classrooms. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635570

1 INTRODUCTION

Computing education literature contains studies examining how students transfer knowledge from block-based to text-based programming languages. One such study found little quantitative difference between the assessment scores between students who had prior Scratch block-based programming experience and those that did not [1]. However, the authors did note that students with prior Scratch experience employed Scratch-specific patterns in their text-based programs. This result suggests that some transfer did occur. Likewise, other studies have shown evidence of positive knowledge transfer from Scratch to text-based programming languages [3, 5]. The literature also contains studies which identify instances of positive transfer of knowledge between text-based languages [3, 5, 7] as well as documenting a lack of transfer [10].

Though these examples show considerable efforts, there has been little work regarding how teachers understand the problem of transfer and how they support students' transition from one programming language to another. Those studies that have examined ways of supporting students in transferring their programming knowledge have largely focused on the development of instructional materials [4] or programming environments [6, 9]. We believe a first step to understanding how and when students successfully transfer their programming knowledge is to understand how and when teachers observe transfer occurring in their classrooms, what explicit connections they make between programming languages, and what resources they are using to do so.

2 RESEARCH DESIGN

An overarching aim for this work is to help students from diverse backgrounds build on prior conceptual knowledge as they progress through a CS course sequence that involves multiple programming languages. The project is conducting a longitudinal study of students' progression through such a course sequence. To identify the ways in which students transfer knowledge from one programming language to another, we first need to understand how and when teachers observe transfer of programming knowledge, what explicit connections they make during their CS instruction, and what resources they are using to support such transfer. Therefore, we conducted interviews with middle and high school teachers (grades 6-12) who have experience teaching both block-based and text-based programming languages. The following research questions framed our study:

(1) What is the teacher's experience and coding background? [*Background*]
(2) What examples of transfer have CS teachers witnessed in their classrooms as their students learn a new programming language? [*Perceptions of Transfer during Learning*]
(3) How do CS teachers currently support students in transferring knowledge from one language to the next? [*Supporting Transfer*]

Eight teachers who self-identified as having experience teaching both Scratch and one other text-based language were recruited for

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635570

interviews. The interviews lasted for 1 hour and were conducted virtually via Zoom. The interview recordings were transcribed, and two members of the research team carried out a thematic analysis [2] of the transcripts. After the first pass of identifying themes, the two researchers reached agreement on refined themes that were grouped into subcategories such as *Spontaneous Transfer*, *Productive Transfer Resources*, or *Curricular Supports of Transfer*. This poster will include all subcategories, their descriptions, and examples from the transcribed interviews.

3 PRELIMINARY FINDINGS

Analysis of teacher interviews revealed that when teachers do make explicit connections between two programming languages, it is most often at the beginning of a new course. This depends on whether or not they know their students' prior programming knowledge. This result was only exhibited for three of the eight teachers who taught multiple CS courses and encountered the same students in more than one course along the curriculum pathway. These teachers indicated that they assumed students had no prior programming experience or that their students expressed having no prior programming experience when they were asked in class. However, the other five teachers noted that, most of the time, their instructional approach is to assume students have no prior programming knowledge. One teacher even suggested this was by design as an instructional technique learned from another content area, "*I assume kids usually don't have the prior knowledge because it was a technique I learned from math.*" Identifying students' prior programming exposure appears to be easier when teachers are at smaller schools and teaching multiple CS courses in the curriculum pathway.

Three of the teachers we interviewed also revealed that they primarily focus on differences in syntax when they are making explicit connections between multiple programming languages. They indicated that these connections were generally in the conceptual context of loops and other control structures, variables, and data structures. One teacher also suggested that they had to make their own classroom resources to show these connections, "*I literally had a PowerPoint and I broke it down to like, Hey, here's one major difference between the two.*"

Finally, four of the teachers interviewed noted that their students sometimes only received instruction in one programming language. These teachers explained that this is a result of students simply fulfilling a credit requirement for their particular curriculum pathway and/or to graduate. However, they perceived this as an issue that could impact knowledge transfer between programming languages because a student might take Scratch during their middle grade years (6-8) and not take another CS course until later in their high school course of study (grades 11-12).

4 CONCLUSION & FUTURE WORK

The study presented in this poster revealed important avenues for moving toward supporting students' programming knowledge transfer via teacher interviews. First, an overall lack of explicit curricular connections between multiple programming languages due to teachers' assumed or perceived potential of students having little or no prior programming experience appears to be at issue. This is an issue that most core academic disciplines do not experience as teachers have an understanding of students' prior knowledge from their course progressions. This strongly suggests a need for a standardized, pedagogically useful learning trajectory for every student on the CS curriculum progression. Further, development of such a learning trajectory could move the needle toward CS being a core discipline in every state with requirements at every grade level.

Additionally, when teachers do make explicit curricular connections between programming languages they tend to largely focus on syntactic differences rather than deeper conceptual knowledge. We believe that this is a missed opportunity for teachers to draw on and take advantage of the prior programming knowledge that their students bring to the classroom. For example, instructional guidance that employs tools like analogical theory [8] could prove to help conceptual knowledge transfer well. The next steps in this project will focus on avenues such as these to drive our work toward a CS-specific theory of knowledge transfer between programming languages. We will then use that theory to implement instructional scaffolds to aid teachers in making more explicit connections and garner the transfer of deeper conceptual knowledge.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation through the EHR Core Research in STEM Learning and Learning Environments program (Award #2201209). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Michal Armoni, Orni Meerbaum-Salant, and Mordechai Ben-Ari. 2015. From scratch to "real" programming. *ACM Transactions on Computing Education (TOCE)* 14, 4 (2015), 1–15.

[2] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[3] Marcos J Gomez, Marco Moresi, and Luciana Benotti. 2019. Text-based programming in elementary school: a comparative study of programming abilities in children with and without block-based experience. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. 402–408.

[4] Shuchi Grover. 2021. Teaching and assessing for transfer from block-to-text programming in middle school computer science. *Transfer of learning: Progressive perspectives for mathematics education and related fields* (2021), 251–276.

[5] Shuchi Grover, Roy Pea, and Stephen Cooper. 2015. Designing for deeper learning in a blended computer science course for middle school students. *Computer science education* 25, 2 (2015), 199–237.

[6] Michael Kölling, Neil CC Brown, and Amjad Altadmri. 2015. Frame-based editing: Easing the transition from blocks to text-based programming. In *Proceedings of the Workshop in Primary and Secondary Computing Education*. 29–38.

[7] Monika Mladenović, Žana Žanko, Andrina Granić, and Andrina Granić. [n. d.]. Mediated transfer: From text to blocks and back. 29 ([n. d.]), 100279. https://doi.org/10.1016/j.ijcci.2021.100279 MAG ID: 3134228633.

[8] Dario D Salvucci and John R Anderson. 2001. Integrating analogical mapping and general problem solving: the path-mapping theory. *Cognitive Science* 25, 1 (2001), 67–110.

[9] Nischal Shrestha, Titus Barik, and Chris Parnin. 2018. It's like python but: Towards supporting transfer of programming language knowledge. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 177–185.

[10] David Weintrop and Uri Wilensky. [n. d.]. Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. 142 ([n. d.]), 103646. https://doi.org/10.1016/j.compedu.2019.103646 MAG ID: 2964965801.

Received 13 October 2023; Accepted 13 November 2023

Enhancing Formative Feedback at Scale with the Intelligent Feedback Assistant

Rifa Jamal
z5311190@unsw.edu.au
University of New South Wales
Sydney, NSW, Australia

Jake Renzella
jake.renzella@unsw.edu.au
University of New South Wales
Sydney, NSW, Australia

ABSTRACT

Formative feedback spans various domains, from education to businesses and creative endeavours. In educational contexts, feedback enriches students' learning and work quality through reflection. However, providing effective feedback at scale is challenging. Students struggle to engage with feedback, often due to lack of feedback literacy. Recent advancements in Natural Language Processing, a branch of Artificial Intelligence, provides opportunities to evaluate how we can support feedback providers in its quality and scale. This poster paper presents an overview of key feedback challenges, attributes of high quality feedback, and introduces the *Intelligent Feedback Assistant (IFA)*, an innovative NLP-based system designed to assist educators in delivering high-quality feedback. IFA operates as an ensemble of machine learning models and non-AI systems to guide educators in refining their feedback, ensuring it embodies attributes of effective feedback — actionable, specific, justified, and positive. IFA is supportive, not generative, ensuring the feedback provider remains central to the feedback provision process. The tool design, and outcomes of IFA offers a promising path for scaleable, high-quality formative feedback in education and beyond.

CCS CONCEPTS

• **Applied computing** → *Computer-assisted instruction*; • **Human-centered computing** → **Natural language interfaces**.

KEYWORDS

Student Feedback, Natural Language Processing, AI in Education

ACM Reference Format:
Rifa Jamal and Jake Renzella. 2024. Enhancing Formative Feedback at Scale with the Intelligent Feedback Assistant. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635482

1 INTRODUCTION

Feedback is crucial for guiding student learning and improving their work quality [4]. However, generating high-quality, effective feedback is a multifaceted challenge, for both new and experienced tutors, as they may lack training in giving effective feedback [7].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635482

Firstly, feedback, subjective to the provider, varies significantly in its content, tone, and utility [5, 9]. Students frequently encounter feedback that is vague, overly generalised, and absent of actionable steps, limiting their improvement [3]. Secondly, feedback without clear justification or reasoning can be perceived as arbitrary or biased, leading to mistrust and confusion among students [5]. Additionally, a lack of positive and constructive tone may discourage students and hinder their receptiveness to feedback [9, 11].

To address these challenges and improve feedback quality for meaningful student progress, four key attributes were identified from literature: *Actionable*, *Justified*, *Specific*, and *Positive tone* [3].

Challenges to meet these attributes are intensified when delivering feedback large scale with high student-to-educator ratios [6]. While human moderation of feedback maintains quality, it is impractical due to the sheer volume of feedback. Continuous training and support mechanisms for educators, both new and experienced, are also essential, but time constraints remain a substantial barrier [10].

While existing Artificial Intelligence (AI) tools including Adaptive Learning Platforms, Intelligent Tutoring Systems, and Large Language Models like ChatGPT offer potential avenues to enhance feedback quality and scalability [2], their use raises concerns.

Educators may hesitate to adopt AI tools, due to the initial learning curve [10]. The necessity of maintaining a human touch in education is emphasised, as an over-reliance on AI could impact interpersonal relationships and the personalisation inherent in effective teaching [6]. Students perform better in response to human written feedback as compared to computer generated feedback [8].

Data privacy and ethical considerations such as bias arise from AI's widespread use in learning, hence the importance for protecting student data and responsible use [2]. These concerns reinforce the reason to keep IFA as supportive, as compared to generative.

Drawing from literature on feedback criteria, challenges and AI in education, we present the Intelligent Feedback Assistant (IFA). IFA employs an ensemble of BERT-based models (Bidirectional Encoder Representations from Transformers), and rule-based systems to provide educators insights on their feedback. Thus, IFA ensures students receive actionable, justified, specific, and positively framed feedback, which is instrumental in their learning journey.

IFA is presented as an open-source API, publicly available free of charge, allowing the tool to be incorporated into any platform. As IFA operates as an evaluative tool, it doesn't generate content. IFA simply assesses formative feedback written by tutors before reaching students, ensuring the human touch remains. The API is intended to be offered no-cost to the wider community.

The architecture of IFA, as shown in Figure 1 incorporates the following models returning assessments in a JSON format:

- The Actionability Model leverages a finely tuned RoBERTa model. It is trained on a diverse dataset comprising of IMDB movie critiques, devoid of calls to action, and a dataset of WikiHow instructions, which naturally contains clear calls to action.
- The Tone Sentiment Model utilises DistilBERT to classify the feedback's tone using an off the shelf model.
- The Justified Model is a preliminary approach, examines the feedback for key compound words to determine if it offers rationale.
- The system processes data locally, without sending to cloud.

1.1 Initial Evaluation and Results

IFA's efficacy was evaluated using real blind-tagged feedback from experienced educators, encompassing 20 samples from research paper reviews and a Computer Science course titled 'Database Fundamentals'. As seen in Figure 1, the IFA interface takes a JSON input (representing feedback to be assessed) from a manual data input into the backend. The 3 machine learning models as described in section 1 then assess and process the feedback. For each sample, IFA returns results structured in JSON, detailing the presence of each criterion as a Boolean value, alongside a confidence score.

To benchmark IFA, it was compared against two off-the-shelf Large Language Models: ChatGPT version 3.5 and Google Bard version 2023.07.13. A prompt was first created to send for each feedback sample consisting of a short rubric, explaining each criteria as well as examples of feedback that matched a specific criteria.

Initial results showed IFA slightly outperformed the LLMs, even without fine-tuning. IFA achieved a score of 70%, slightly leading ChatGPT's 65% and closely trailing Bard's 75%. When examining justifiable feedback, IFA took the lead with 70%, compared to ChatGPT's 40% and Bard's 60%. IFA and Bard stood equal in terms of feedback tone, each scoring 65%, and ChatGPT at 60%.

Despite having rubric-based prompts, discrepancies emerged with ChatGPT and Google Bard. They often contradicted the evaluation of the same feedback provided in the rubric as examples. Results resembled a case study revealing ChatGPT had low false negatives (170), but high false positives (1,118) [8], highlighting LLM's shortcomings to interpret and apply rubrics and examples [8].

2 FUTURE WORK

The promising results from IFA indicate that by fine tuning IFA with real, tagged feedback, we can further improve its efficiency.

IFA bridges technological gaps in education; rather than overshadowing educators, it leverages AI to help educators provide feedback more efficiently by training systems to behave in desirable ways. This is known as 'Scaling Supervision' Bai et al. [1] and helps keep the level of human touch. The initial research has paved the path for planning further development of the IFA system.

3 CONCLUSION

The integration of AI tools in education offers promising avenues for efficient and effective feedback, but raises significant challenges such as technical experience, scale, data privacy and more. This poster establishes the potential for AI-driven feedback systems, particularly IFA, to navigate these challenges and improve education outcomes while maintaining privacy and educators' roles.

By utilising BERT-based language models, IFA strives to analyse feedback before it is provided to students via four criteria: actionable, positive tone, justified, and specific. With its design as an open-source API, continued development and research will aim to achieve a seamless integration of IFA in the teaching space. Examples include integration with current feedback systems or creating it as a browser extension. It will be an effective complement to human educators, through improvement of educational feedback and enhancing learning experiences.

REFERENCES

[1] Yuntao Bai et al. 2022. Constitutional AI: Harmlessness from AI Feedback. arXiv:2212.08073 [cs.CL]

[2] Lijia Chen, Pingping Chen, and Zhijian Lin. 2020. Artificial Intelligence in Education: A Review. *IEEE Access* 8 (2020), 75264–75278. https://doi.org/10.1109/ACCESS.2020.2988510

[3] John Hattie and Helen Timperley. 2007. The Power of Feedback. *Rev Educ Res* 77 (2007), 81–112.

[4] Ende J. 1983. Feedback in clinical medical education. *Journal of the American Medical Association* 250(6) (1983), 777–781.

[5] Rachel Jug, Xiaoyin Sara Jiang, and Sarah M. Bean. 2019. Giving and Receiving Effective Feedback: A Review Article and How-To Guide. *Archives of pathology & laboratory medicine* 143 (2019), 244–250. Issue 2. https://doi.org/10.5858/ARPA.2018-0058-RA

[6] Alwyn Vwen Yen LEE. 2023. Supporting students' generation of feedback in large-scale online course with artificial intelligence-enabled evaluation. *Studies in Educational Evaluation* 77 (6 2023), 101250. https://doi.org/10.1016/J.STUEDUC.2023.101250

[7] Fritz Lekschas, Spyridon Ampanavos, and Pao Siangliulue. 2021. Ask me or tell me? enhancing the efectiveness of crowdsourced design feedback. *Conference on Human Factors in Computing Systems - Proceedings* (5 2021). https://doi.org/10.1145/3411764.3445507

[8] Huy A Nguyen, Sarah Di, and Bruce M Mclaren. 2023. Evaluating ChatGPT's Decimal Skills and Feedback Generation in a Digital Learning Game. (2023).

[9] Thi Thao Duyen T Nguyen, Thomas Garncarz, Felicia Ng, et al. 2017. Fruitful Feedback: Positive affective language and source anonymity improve critique reception and work outcomes. In *Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing*. 1024–1034.

[10] Naomi Winstone and David Carless. 2019. Designing Effective Feedback Processes in Higher Education: A Learning-Focused Approach. *Designing Effective Feedback Processes in Higher Education: A Learning-Focused Approach* (1 2019), 1–208. https://doi.org/10.4324/9781351115940

[11] Lu Xing, Jian min Sun, and Denise Jepsen. 2021. Feeling shame in the workplace: Examining negative feedback as an antecedent and performance and well-being as consequences. *Journal of Organizational Behavior* 42 (11 2021), 1244–1260. Issue 9. https://doi.org/10.1002/JOB.2553

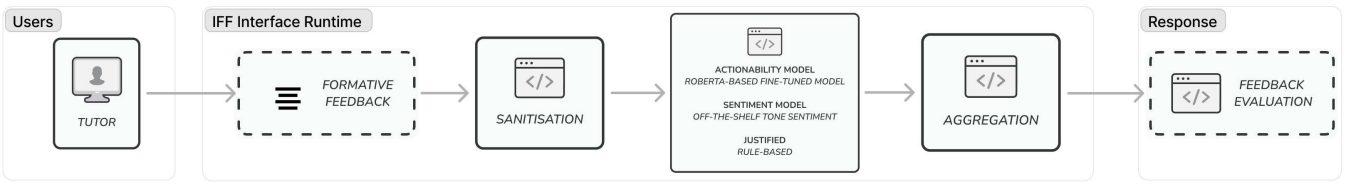


Figure 1: Diagram of the IFA inference pipeline, depicting tutor-authored feedback aggregated into a final evaluation.

Striking a Chord through a Mixed-Methods Study of Music-Based Learning to Leverage Music and Creativity to Bridge the Gender Gap in Computer Science

Fatemeh Jamsidi
Computer Science and Software Engineering
Auburn University
Auburn, AL, USA
fzj0007@auburn.edu

Maryam Bigonah
Computer Science and Software Engineering
Auburn University
Auburn, AL, USA
mzb0197@auburn.edu

Daniela Marghitu
Computer Science and Software Engineering
Auburn University
Auburn, AL, USA
marghda@auburn.edu

ABSTRACT

The persistent gender gap in technical fields, particularly in computer science (CS), remains a complex challenge. This paper draws upon recognized gender differences in aptitude, interest, and the educational environment. It builds upon established theoretical foundations, including Bandura's Social Cognitive Theory [9] and Eccles's Expectancy-Value Theory [3][2], to explore innovative strategies for engaging underrepresented groups, specifically female students, in CS.

Central to our approach is an innovative curriculum that integrates the realms of math, computing, and music. Recognizing the intrinsic connection between music and mathematical patterns, our pedagogical model simplifies complex CS concepts, transforming topics like looping, algorithms, and conditional statements into engaging, relatable lessons. Additionally, our curriculum incorporates foundational elements of robotics, enhancing its practical applicability and real-world relevance.

This groundbreaking educational paradigm not only simplifies technical concepts but also leverages students' innate passion for music, creating an intuitive and enjoyable learning environment. By blending artistic creativity with scientific precision, this approach sets a new standard for curriculum design and offers educators a pioneering roadmap to address the gender gap in CS. Our study aligns with SIGCSE's mission, marking a fresh, inclusive trajectory for the future of technology education.

CCS CONCEPTS

• **Applied computing** → **Collaborative learning**; **Computer-assisted instruction**; **Interactive learning environments**; • **Human-centered computing** → **Collaborative and social computing**; *Human computer interaction (HCI).*

KEYWORDS

K-12 Computer Science Education; Creativity; Computer Music; STEAM; Curriculum Design; EarSketch

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635612

ACM Reference Format:
Fatemeh Jamsidi, Maryam Bigonah, and Daniela Marghitu. 2024. Striking a Chord through a Mixed-Methods Study of Music-Based Learning to Leverage Music and Creativity to Bridge the Gender Gap in Computer Science. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635612

1 INTRODUCTION

Theoretical Foundation: Our research aligns with Bandura's emphasis on self-efficacy and individual agency, intertwined with Eccles's insights into personal values, expectations, and engagement in learning. Guided by these frameworks, we investigate how personal expression, creativity, and aesthetics can stimulate engagement in technical domains [5].

Research Questions:

- How does a CS + Music environment, emphasizing personal expression and creativity, impact the engagement and persistence intentions of underrepresented groups, particularly girls?
- What predictive relationships exist between a leadership-authentic learning environment and increased intentions to persist in CS, mediated by factors like identity and creativity?
- How can the adaptation of tools like Flowkey, Melodics, EarSketch, and Scratch in secondary-school-level programming education inspire a more inclusive and stimulating learning environment?

In this mixed-methods study, we introduce computer music as a unique avenue to connect students with CS. We share our experiences in adapting educational tools for an introductory programming course at a summer camp held at Auburn University, where American girls aged 12 - 16 actively participated in extensive surveys.

A quasi-experimental component of our study highlights leadership-authentic learning environments, revealing connections that predict increased intentions to persist in CS, influenced by factors such as identity and creativity. We also conducted a path analysis to further explore elements related to student engagement and intentions to continue in CS.

Our findings underscore the vital role of a compelling CS + Music experience in K-12 classrooms for enhancing confidence and interest among underrepresented groups [10]. By bridging computer science with the arts and emphasizing creativity and personal

expression, this study offers valuable insights and actionable strategies to help close the gender gap in technical fields. It charts a course for educators, policymakers, and researchers, guiding them towards a more inclusive and expressive future in technology education.

2 METHODS

2.1 Pilot Population and Learning Context

One of the research cases, the Mentoring Alabama Girls in Computing (MAGIC) camp, was piloted with 24 middle school and high school students as a computing and music camp at Auburn University. The 1-week pilot took place from June 16th to June 20th, 2019.

The MAGIC curriculum includes EarSketch and Scratch, with the addition of leadership concepts, marking the first curriculum in a three-camp computing pathway - Robotics, App Development, and CS for all girls programs in Auburn and surrounding communities. Its curriculum covers programming and CS principles, along with modules on leadership and music remixing. The EarSketch and Scratch pilot served as the programming module of the course.

Many programs worldwide have aimed to increase awareness of STEAM education for female students, such as "Girls, Music, and Computer Science" [1] [7] and "Music Education Meets Computer Science and Engineering Education" [8]. In our efforts to attract underrepresented female students to future careers in CS and Music, we developed the MAGIC + M program - a curriculum combining music, CS, and leadership. Within this project's context, we believe that Perfomatics is a potential area that may attract students to CS. The main steps of the curriculum include:

- Introducing students to CS concepts and skills using Scratch music-related games.
- Introducing students to music-based storytelling.
- Teaching the basics of music programming using EarSketch.
- Introducing students to basic concepts of leadership.
- Teaching how to generate their musical compositions.
- Providing an opportunity for student presentations in a music composition competition.

3 EVALUATION

In this pilot study, we designed qualitative and quantitative surveys to measure attitude changes and content knowledge among students. A pre/post retrospective student engagement survey was administered at the beginning and end of the pilot. The survey asked students to reflect on their feelings about the course and their background in music and computer science concepts.

The instrument used in this project is based on "Engaging Underrepresented Groups in High School Introductory Computing through Computational Remixing with EarSketch," in which we measure six psychosocial constructs:

(1) Computing confidence.
(2) Computing enjoyment.
(3) Computing importance/perceived usefulness.
(4) Motivation to succeed [6].
(5) Identity and belonging in computing.
(6) Creativity in computing [4].

The fifteen-question survey was designed to demonstrate changes in these six psychosocial constructs after students attended the program activities.

Our analysis includes a paired samples t-test to identify significant changes from pretest to post-test.

REFERENCES

[1] Ali Alshammari. 2023. Examining the Relationship Between Components of the MUSIC Model of Motivation and Student Achievement in Computer Programming. *International Journal of Learning, Teaching and Educational Research* 22, 1 (jan 30 2023), 283–301.
[2] Cheryl Beauchamp and Holly Matusovich. 2023. A Mixed-Method Study Exploring Cyber Ranges and Educator Motivation. *Journal of Cybersecurity Education Research and Practice* (2023). https://api.semanticscholar.org/CorpusID:266511183
[3] Chen Chen, Jonathan Rothwell, and Pedrito Maynard-Zhang. 2023. In-school and/or out-of-school computer science learning influence on CS career interests, mediated by having role-models. *Computer Science Education* (2023). https://api.semanticscholar.org/CorpusID:266581115
[4] Ramona Mariana Cojocaru-Spînu et al. 2022. MOTIVATION AND CREATIVITY IN WORKING WITH POP/ROCK/JAZZ BANDS OF SECONDARY AND HIGH SCHOOL LEVEL. *Review of Artistic Education* 23 (2022), 81–86.
[5] Ville Isomottonen, Antti-Jussi Lakanen, and Paavo Nieminen. 2020. Exploring Creativity Expectation in CS1 Students' View of Programming. In *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE.
[6] Christian Köppe. 2020. Program a Hit – Using Music as Motivator for Introducing Programming Concepts. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. ACM.
[7] Veronica Madrigal, Ryoko Yamaguchi, Adam Hall, and Jamika Burge. 2020. Promoting and Supporting Computer Science Among Middle School Girls of Color. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM.
[8] Tom McKlin, Lauren McCall, Taneisha Lee, Brian Magerko, Michael Horn, and Jason Freeman. 2021. Leveraging Prior Computing and Music Experience for Situational Interest Formation. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. ACM.
[9] Welly Suryandoko. 2023. Bandura's Social Cognitive Theory and Its Implementation in Markeso Mudo's Garingan Ludrukan Monologue. *Gondang: Jurnal Seni dan Budaya* (2023). https://api.semanticscholar.org/CorpusID:263844762
[10] Ryoko Yamaguchi, Veronica Hankerson Madrigal, Cyntrica N. Eaton, and Jamika D. Burge. 2023. Equitable STEM+CS learning experiences for girls of color: nurturing an independent learning approach via a learning ecosystem. *Journal for Multicultural Education* (jul 4 2023).

Root the (Ballot) Box: Designing Security Engineering Courses with E-Voting

Tushar M. Jois
City College of New York
New York, NY, USA
tjois@ccny.cuny.edu

Atheer Almogbil
Johns Hopkins University
Baltimore, MD, USA
aalmogb1@jhu.edu

Logan Kostick
Johns Hopkins University
Baltimore, MD, USA
lkostic1@jhu.edu

ABSTRACT

Security courses often focus on individual attacks and defenses and lack practical experience with secure system design. Consequently, such courses rarely address the societal implications of security breaches. We propose an inductive teaching approach to address these challenges in teaching security engineering. To this end, we designed a course that focuses on electronic voting (e-voting) as a practical application of security engineering. Our course integrates relevant technical content through experiential learning, adversarial thinking skills through a group project, and societal factors with a student debate. Our approach has seen success in engaging students in both distance-learning and in-person teaching while preparing them for real-world security challenges.

ACM Reference Format:
Tushar M. Jois, Atheer Almogbil, and Logan Kostick. 2024. Root the (Ballot) Box: Designing Security Engineering Courses with E-Voting. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635518

1 BACKGROUND & RELATED WORK

Computer security courses teach a standard set of attacks and defenses, but these are often too abstract for students to understand. There has been some progress in this regard, with materials like SEED [3] providing hands-on learning activities on the attacks and defenses of computer security. These activities, however, do not encompass security *engineering*, the design and analysis of secure systems. A key component of security engineering is *adversarial thinking*, in which students attempt to identify potential threat actors in a system and their capabilities. Experiential introductions to security concepts, while important, miss this requirement. Moreover, security engineering necessitates understanding the *social context* of the system and the impacts of a security breach. These impacts are personal; all users – including students – are at risk.

Prior work contains examples of curricula that incorporate real-world applications where students feel like their learning is immediately relevant (e.g., [2, 5, 7, 10]). What is missing, however, is a *concrete* problem that encourages adversarial thinking while being realistic enough to motivate course material. Electronic voting (e-voting) system design is a security engineering problem that captures both dimensions. E-voting does away with the paper ballots traditionally associated with elections, replacing both vote casting and tabulation with a computer. Early systems were vulnerable to a litany of attacks, generating academic interest in analysis [6]. At the same time, the problem has major political implications [4].

We propose a new type of course that utilizes e-voting to teach security engineering, combining technical content, adversarial exercises, and societal connections to successfully prepare students for the future. As with a traditional course, we aim to give students an understanding of different attacks and defenses. Our course also has students engage with the social implications of computer security, as well as give them experience in how security analyses are done in practice. We do this by integrating two inductive teaching approaches [9]: project-based learning to build adversarial thinking, and case-based teaching to contextualize the course material.

Contributions. To our knowledge, this work represents the first investigation into security engineering education through such a method. Our contributions are as follows:

- We describe the content for a new type of security engineering course that utilizes e-voting as an application.
- We provide our experiences and lessons learned running the proposed course via both distance and in-person learning.
- We discuss future directions for this curriculum design.

2 APPROACH

In developing the course, we adhered to the following design principles: the course should (1) contain the required technical content for a security course, (2) build adversarial thinking skills, and (3) incorporate societal dimensions. Additionally, since we initially designed the course during the COVID-19 pandemic, we wanted the course flexible enough to support distance learning if required. As such, we ensured that none of these proposed course elements require a specific course delivery method.

Technical content. We aim to provide the same technical content as an equivalent traditional cybersecurity course. We use the SEED Labs [3], a suite of hands-on computer security labs, to provide this. The labs cover software, network, web, systems security, and an introduction to cryptography. Because the labs are performed on a virtual machine, the students (and instructors) do not have to worry about damaging a production system and can ensure all coursework is performed in a standard environment.

These labs have been successfully used in prior versions of the security and privacy course at Johns Hopkins University. But, since our new course must accommodate additional security engineering-related material, we must get through this technical content quicker than a usual course. To mitigate this increased pace, we add more

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635518

support and guidance for students in the form of recitation sections. During these sections, teaching assistants guide students through the finer details of the labs and assist with any issues.

Adversarial thinking. To build adversarial thinking skills, we utilize project-based learning [9], in the form of a group project that allows students to creatively apply their learning to a problem.

First, in the *blue team* phase, groups of 2–3 students design an e-voting machine that is *intentionally* vulnerable to several of the attacks they learned in the course. Then, in the *red team* phase, each group receives a different voting machine and has to identify and exploit as many vulnerabilities as possible. This approach is similar to "root-the-box" or "capture-the-flag"-type games (e.g., [8]), but we add a voting machine application component that motivates the assignment. At the end of the course, students present their (blue team) vulnerable machine design and the vulnerabilities identified in the given (red team) machine to the rest of the class.

We provide starter code to the students for the blue team phase that they may use for the project. Our starter implementation – a networked e-voting system with a web application interface – was set up to match the technical themes of the SEED Labs. This structure allows students to apply all of their knowledge in a structured environment. The friendly competition inherently creates adversarial thinking skills, with students taking the perspective of a malicious actor in developing the vulnerable voting machine. The students must understand what makes a system vulnerable before they are able to inject intentional vulnerabilities. Moreover, the project gives students practice in penetration testing, a highly sought-after skill in industry [1] where this blue-team–red-team approach to security engineering is common.

Societal dimensions. We also want students to get out of classroom experience and see how their security work impacts the real world. To this end, we split the class into two teams and had them debate and discuss the merits of e-voting and its impact on society. Because the debate is straight out of the headlines, students may have already been exposed to (and perhaps interested in) these topics. We introduce e-voting's impact on society through a lecture on its history, from the 2000 U.S. election to present. Then, we have each team collaborate on the points they wish to present and defend. Since e-voting has elicited strong opinions in the media, we encourage students to perform research using trustworthy sources and defend their claims with evidence. The debate is then held in class, with students voting on which team did the best.

3 INITIAL PILOTS

We first implemented our course as the introductory security & privacy course at Johns Hopkins University in Fall 2020. Due to the COVID-19 pandemic, we taught this course via distance learning. Students responded positively to the course, despite the challenges inherent to online learning, finding that the course structure kept them engaged throughout the semester. We also ran the course in Fall 2021 and Fall 2022 but in an in-person setting instead. Because of the flexibility of our curriculum design, the course translated well to in-person learning, and we received similar positive feedback.

We believe that the success of the course stems from its focus on project-based learning and case-based teaching, which allows students to take charge of their own learning [9]. Moreover, the collaborative nature of the project and debate encourages students to interact with each other. This type of collaboration is natural and common in-person, but can be challenging to foster online.

We aim to have other courses to apply our approach to a security engineering curriculum. To this end, all of our course materials are available at https://github.com/spacelab-ccny/rtbb.

4 CONCLUSION & FUTURE WORK

We present a new type of security engineering course centered around e-voting. It has the same technical content as a traditional course but also builds adversarial thinking skills in students and incorporates societal connections into its teaching. We describe its structure and provide our thoughts on running the course in distance and in-person learning environments.

We will continue to iterate on our course design, incorporating the lessons we learned over time. For example, our course could integrate more security engineering themes, such as the uniquely human elements of social engineering and physical security. To add this content, we plan to add a simulated "election day," in which some students act as poll workers and voters while others attempt to subvert the election. Students could then run their technical exploits on physical machines after bypassing human controls. This would crystallize the notion of an insider threat while being interactive and engaging. This could also be done in a distance learning setting, with local voting clients and a remote tabulation server. Additionally, our evidence so far is anecdotal. We desire to run a more quantitative study in the future, collecting feedback and running focus groups to review the design of the course.

ACKNOWLEDGMENTS

The authors would like to acknowledge support from the NSF under award 1955172. The views contained herein are those of the authors and should not be interpreted as those of the NSF. The authors would also like to thank Avi Rubin and Max Zinkus for their contributions to the development of the course.

REFERENCES

[1] Miriam E Armstrong, Keith S Jones, Akbar Siami Namin, and David C Newton. 2018. What vulnerability assessment and management cybersecurity professionals think their future colleagues need to know. In *SIGCSE '18*.
[2] Ruedi Arnold, Marc Langheinrich, and Werner Hartmann. 2007. InfoTraffic: Teaching Important Concepts of Computer Science and Math through Real-World Examples. *SIGCSE Bulletin* 39, 1 (mar 2007), 105–109.
[3] Wenliang Du and Ronghua Wang. 2008. SEED: A suite of instructional laboratories for computer security education. *Journal on Educational Resources in Computing (JERIC)* 8, 1 (2008), 1–24.
[4] Andrew C Eggers, Haritz Garro, and Justin Grimmer. 2021. No evidence for systematic voter fraud: A guide to statistical claims about the 2020 election. *Proceedings of the National Academy of Sciences* 118, 45 (2021), e2103619118.
[5] Mounib Khanafer and Tushar M. Jois. 2023. Towards Application-Driven IoT Education. In *EDUCON '23*. IEEE.
[6] Tadayoshi Kohno, Adam Stubblefield, Aviel D Rubin, and Dan S Wallach. 2004. Analysis of an electronic voting system. In *IEEE Symposium on Security & Privacy*.
[7] Daniel E. Krutz, Samuel A. Malachowsky, and Thomas Reichlmayr. 2014. Using a Real World Project in a Software Testing Course. In *SIGCSE '14*. 49–54.
[8] Jelena Mirkovic and Peter AH Peterson. 2014. Class Capture-the-Flag Exercises. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education*.
[9] Michael J Prince and Richard M Felder. 2006. Inductive teaching and learning methods: Definitions, comparisons, and research bases. *Journal of engineering education* 95, 2 (2006), 123–138.
[10] Sarah J. Van Wart, Sepehr Vakil, and Tapan S. Parikh. 2014. Apps for Social Justice: Motivating Computer Science Learning with Design and Real-World Problem Solving. In *ITiCSE '14*. 123–128.

Bridging the Gap: Computational Thinking as a Pathway to Equitable Math Education for Neurodiverse Learners

Kate Kastelein
Maine Mathematics and Science Alliance
Augusta, Maine
kkastelein@mmsa.org

Kelsie Dawson, PhD
Maine Mathematics and Science Alliance
Augusta, Maine
kdawson@mmsa.org

Rhonda Tate
Maine Mathematics and Science Alliance
Augusta, Maine
rtate@mmsa.org

ABSTRACT

Students who receive special services in mathematics or neurodiverse learners (NDLs) deserve to learn mathematics at a rigorous and equitable level as their peers. Yet, many of these students are not reaching proficiency. Computational thinking (CT) may function as a bridge between students' mathematical learning and their everyday lives. We argue that computational thinking is a pathway for NDLs to follow that will bridge the connections between math instruction and the real-life problems with which NDLs are eager to engage. We identified the critical elements needed to support rural special educators' integration of CT into math instruction by identifying intersections between CT, Math, and the assets of NDLs, as well as barriers to implementation. By leveraging connections from an existing RPP (Research-Practice Partnership), which focuses on CS education and Business intersection in rural Maine schools, we created an opportunity for participatory research for special educators and paraprofessionals often overlooked in these processes. We employed a multi-method, design-based research (DBR) approach. Through observational, survey, and interview data, we found that 1) educators are not currently receiving support for integrating CT in special education classrooms, 2) activities piloted by our educators were well received by students, 3) current standards and Individualized Educational Plan (IEP) requirements for NDLs are a significant barrier to implementation. The need for expanded, regular PD for special educators is evident, as is the potential for CT as a bridge for mathematical understanding for NDLs.

ACM Reference format:
Kate Kastelein, Kelsie Dawson and Rhonda Tate. 2024. Bridging The Gap: Computational Thinking as a Pathway to Equitable Math Education For Neurodiverse Learners In *Proceedings of ACM SIGCSE conference (SIGCSE '24). ACM, New York, NY, USA, 2 pages.*
https://doi.org/10.1145/3626253.3635476

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635476

1 INTRODUCTION

The identification of intersections and barriers present in the use of an asset-based approach for the implementation of Computational Thinking (CT) in math instruction for Neurodiverse Learners (NDLs) is a vital first step in understanding viability of this model. Due to the frequent use of a deficit-based approach when it comes to educating NDLs, particularly in the instruction of mathematics, where the practice of rote memorization is still commonplace [1, 2] NDLs may not have the same opportunities to benefit from CT instruction as their peers. This is despite evidence that indicates NDLs are often uniquely gifted in many of the areas that define CT, such as abstraction, pattern recognition, and decomposition [1,3]. Historically, professional learning for effective instructional practices in mathematics for Special Educators has been limited [4]. While paraprofessionals have become an increasingly integral part of delivering special education services, many have not received professional development and support in evidence-based instructional strategies in math [5] or in CT-integrated approaches. Paraprofessionals, those who work most frequently with NDLs, are often left out completely from these forms of research-based professional learning, even though they work most closely with NDLs and know their strengths better than others in the school building. This situation playing out in schools across the nation generates a disproportionately inequitable and potentially harmful scenario for SEs, paraprofessionals, and the NDLs themselves.

2 PROGRAM

Our program brought together educators to engage in and discuss using computational thinking as a mechanism to further increase NDLs' engagement using an asset-based framework. The project used an RPP process to determine initial ideas around implementing CT-integrated math lessons and the identification of professional learning supports necessary for rural districts to utilize in the integration of computational thinking in the special education curriculum with a specific focus on the intersection of Math and CT Practices with the strengths of NDLs.

On January 6th, 2023, we convened a group of 13 individuals from backgrounds in special education, school administration, math instruction, and CT/CS (Computational Thinking and Computer Science). The goal of this workshop was to introduce participants

to key concepts relating to asset-based instruction for neurodiverse learners, math practices, and computational thinking and then find intersections where the three areas align. The focuses of the workshop were:

- Introduce participants to key concepts in each of the concentration areas, Computational Thinking, Math Practices, and Asset-Based approach to teaching neurodiverse learners
- Provide examples of CT activities
- Identify intersections of CT, Math and NDL assets
- Identify barriers to integration of an asset-based approach of teaching CT to NDLs

After the introduction of key-concepts participants were given post-it notes and asked to write their thoughts about where intersections occurred and to place them on a diagram (Fig 1). Researchers documented the placement of post-its and analyzed the results.

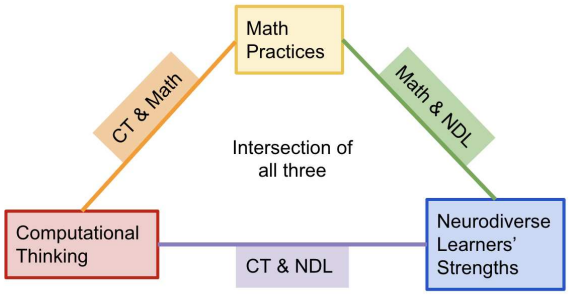


Figure 1: Diagram participants used to identify intersections of CT, Math, and NDL strengths. Each vertex is labeled with one area of intersection study, and the center of the triangle represents where all three intersect.

2.1 Method and Findings

Using emergent coding [6] of the post-its and the documented placement on the chart, we found that, out of 109 responses recorded on post-it notes, 59 were placed in the intersection of all 3 areas of the diagrams. Abstraction, persistence, creativity, decomposition, and pattern recognition were the most often mentioned examples of intersections between CT, Math, and the strengths of NDLs.

We have learned there is energy and excitement around providing NDLs access to problem-based mathematics through a CT lens. Our preliminary findings indicate that educators, administrators, and support staff are eager to continue this work. Early survey analysis finds that the RPP format allows participants, regardless of career level, to feel that their voices are heard and contributions are valued. When asked to identify barriers to implementation the top three answers were:

- Teacher PD: Understanding/Knowledge
- Lack of Buy-in From All
- IEPs Goals and Mandates

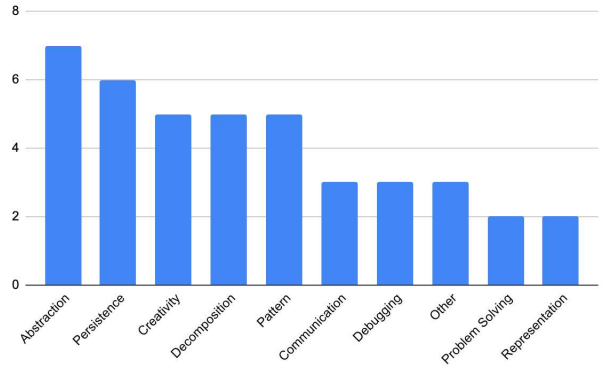


Figure 2: Graph depicting the top 10 most often mentioned areas of intersection of CT, NDLs, and Math practices.

3 DISCUSSION

Despite enthusiasm for creating an asset-based framing approach for integrating CT into math, many participants voiced difficulty in implementing this approach in their classrooms due to IEP and other strict standard requirements when working with this population of students. By presenting this data, we hope to highlight ways in which CT, math practices, and NDL assets intersect to inspire continued thought and discussion around how these practices may be integrated into classrooms.

REFERENCES

[1] Armstrong, T. (2012). Neurodiversity in the classroom: Strength-based strategies to help students with special needs succeed in school and life. ASCD.

[2] Lambert, R. (2018). "Indefensible, illogical, and unsupported"; countering deficit mythologies about the potential of students with learning disabilities in mathematics. *Education Sciences*, 8(2), 72.

[3] Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

[4] Newton, J. R., Kennedy, M. J., Walther-Thomas, C., & Cornett, J. (2012). Transforming Special Education Teacher Education: A Reaction to the Report. *Focus on Exceptional Children*.

[5] Brock, M. E., & Carter, E. W. (2013). A systematic review of paraprofessional-delivered educational practices to improve outcomes for students with intellectual and developmental disabilities. *Research and Practice for Persons with Severe Disabilities*, *38*(4), 211-221.

[6] Miles, M. B., Huberman, A. M., & Saldana, J. (2019). Qualitative data analysis: A methods sourcebook (4th ed.). Sage Publications.

Computer Science Kickstart: An Innovative Bootcamp to Ignite Passion in First-Year Female-Identifying University Students

Stella Kaval
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA USA
stellakaval@berkeley.edu

Mallika Reddy
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA USA
mallikareddy@berkeley.edu

Abigail O'Neill
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA USA
abigator7@berkeley.edu

ABSTRACT

The UC Berkeley CS Kickstart initiative represents a novel approach in addressing the underrepresentation of women in the tech industry. This paper evaluates the efficacy of its one-week intensive curriculum in computer science, particularly focusing on first-year female-identifying university students. The 2023 program cohort witnessed the introduction of a dual-track system, accommodating both beginners and students with prior experience. This study emphasizes the findings from enhanced survey methods and involved detailed pre-program and post-program assessments that gauged the initiative's success. The findings reveal that such targeted, short-term educational programs significantly contribute to advancing diversity and inclusion in computer science education. The results suggest that early intervention at the university level can play a pivotal role in shaping a more diverse future in the tech landscape.

KEYWORDS

Gender equity, short-term programs, first-year female-identifying university students

ACM Reference format:

Stella Kaval, Mallika Reddy, and Abigail O'Neill. 2024. Computer Science Kickstart: An Innovative Bootcamp to Ignite Passion in First-Year Female-Identifying University Students. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635611

1 BACKGROUND

CS Kickstart is a program at UC Berkeley, established in 2011 by Professor Coleen M. Lewis and Floraine Berthouzoz. CS Kickstart is a free one-week residential initiative, held before the orientation week, exclusively for first-year female-identifying college students. The program aims to introduce these students to computer science through various activities such as lectures, programming labs, workshops, talks from faculty, industry field trips, and a full dorm-living experience. The goal is to provide an immersive introduction to computer science and help build a supportive college network for the participants. The latest iteration of CS Kickstart, in its 12th year, occurred in August 2023 with 98 participants. This year marked the introduction of a dual-track curriculum and the implementation of additional participant surveys to gauge the program's effectiveness.

1.1 Curriculum Overview

CS Kickstart's curriculum is divided into several segments, including 6 hours of Python programming labs and 8 hours dedicated to students' individual capstone projects. Additionally, the curriculum features 6 hours of application workshops, including electrical engineering, web development, and data science.

1.2 New Dual-Track Addition

The new dual-track addition created a beginner and advanced version of both the Python programming lab section, as well as the capstone project. Track-level was self-selected by students, and we recommended that students with AP Computer Science A or equivalent experience select the advanced track. Students were grouped into lab sections of 10 same-track students with 2-4 program volunteers. The beginner lab groups assumed zero prior knowledge of code, while the advanced lab groups offered harder practice problems that aligned with the difficulty of the first 2 weeks of the first UC Berkeley computer science course.

As for the capstone project, students chose from web development, Python programming, and data science. Each track had its own example projects with the beginner track offering more optional skeleton code. The projects were tailored to the interest and skill levels of the students, ranging from personal websites to video games, or even data analysis on topics such as forest fire prevention. This segment of the curriculum was supported by a healthy ratio of volunteers to students (approximately 1: 5), ensuring personalized guidance and support.

In previous iterations of the program, students with zero experience felt the material was introduced too quickly, while some advanced students felt bored and unchallenged. With dual-

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635611

track, the initiative is designed to make students at all skill levels feel engaged and excited about continuing their journey in computer science.

2 METHODS

The primary objective of our research was to evaluate the effectiveness of the CS Kickstart program at a large public US university in terms of imparting foundational computer science knowledge and fostering participant enthusiasm and confidence. All participants, spanning both the beginner and advanced tracks, were invited to participate. Data was collected through pre-program and post-program surveys, measuring feedback, skill enhancement, and interest in computer science. We had 98 responses on the pre-program survey and 92 on the post-program survey in August 2023. Both surveys were optional, but highly recommended for students.

3 RESULTS

3.1 Pre-program versus post-program survey

In our recent study evaluating the impact of CS Kickstart on 98 incoming female-identifying students, we observed some compelling findings about its influence. Before participating in CS Kickstart, our preliminary survey revealed a mere 11% of students reported feeling very prepared or confident about entering their first computer science course at the university. However, post-program, there was a dramatic shift in sentiment. A significant 95% of participants reported feeling better prepared to tackle their initial computer science class. Moreover, an overwhelming 96% expressed that their involvement in CS Kickstart positively bolstered their confidence to pursue computer science. In terms of community and belonging, 99% felt they were beginning their freshman year with friends that had similar interests in computer science, and the same percentage believed that the program provided them a clearer understanding of the discipline. Most telling, 100% of the students expressed that they felt comfortable and included amongst their peers and the CS Kickstart staff, signifying a positive learning experience. Aside from these statistics, we also collected demographic information and feedback for each faculty talk, field trip, and activity to help improve future offerings of the program.

3.2 Dual-track testimonials

The program's dual-track system, allowing tailored engagement based on experience levels, emerged as especially beneficial for optimal learning. Feedback from the written responses of the post-program survey affirmed positive results of the dual track system.

> "I think the beginner track pushed me the perfect amount and allowed me to think for myself on difficult problems without being too overwhelmed."

However, participants also pointed out potential areas for enhancement, such as more time to work on projects, additional breaks between work sessions, and more program staff to answer their questions.

4 CONCLUSION

The CS Kickstart program has generated notable positive outcomes among its participants. The data from post-program surveys suggests a robust increase in both confidence and enthusiasm towards computer science. It is not just about teaching the basics of a discipline, but also about fostering a sense of belonging and forging invaluable connections early in a student's academic journey. CS Kickstart stands as a testament to how a short-term, well-structured program can significantly influence and potentially reshape the academic trajectory of female-identifying students in the realm of computer science.

4 FUTURE WORK

As CS Kickstart continues its mission to promote diversity and inclusion in computer science education, several avenues for future work and expansion come into focus. First, we aim to extend the reach of the CS Kickstart program beyond our university. This involves developing partnerships, sharing our curriculum, and offering support to facilitate the implementation of similar programs. In the past year, we have compiled a guide to assist universities in starting their own CS Kickstart program: https://cs-kickstart.gitbook.io/expansion/. Aside from expansion, we plan to generate more data and iterate on our methods of evaluating program success. This includes creating a more comprehensive program survey, as well as quantifying long-term impact of this short-term program through tracking academic performance, retention rates in computer science, and career paths. Last, CS Kickstart will maintain a dynamic curriculum, regularly updating content and teaching methods to align with industry trends. In summary, CS Kickstart remains committed to advancing diversity and inclusion in tech education, with a clear path for future expansion and improvements.

ACKNOWLEDGMENTS

This work was sponsored by the University of California, Berkeley EECS Department and various corporate sponsors. The work was aided by Audrey Sillers, Director of Student Diversity in the UC Berkeley EECS Department and built upon the program initially created by Professor Coleen M. Lewis and Floraine Berthouzoz, the co-founders of the CS Kickstart program. The authors were the lead directors of the 2022 and 2023 CS Kickstart program. All authors contributed equally.

REFERENCES

[1] Nguyen, A. & Lewis, C. M. 2020. Competitive Enrollment Policies in Computing Departments Negatively Predict First-Year Students' Sense of Belonging, Self-Efficacy, and Perception of Department.

[2] Lewis, C. M., Bruno, P., Raygoza, J., & Wang, J. 2019. Alignment of Goals and Perceptions of Computing Predicts Students' Sense of Belonging in Computing. International Computing Education Research (ICER) Conference. 11-19.

A Family of Instruments to Measure Data Science Attitudes

April Kerby-Helm
Department of Mathematics and Statistics, Winona State University, Winona, MN USA
akerby@winona.edu

Michael A. Posner
Department of Mathematics and Statistics, Villanova University, Philadelphia, PA USA
michael.posner@villanova.edu

Alana Unfried
Department of Mathematics and Statistics, California State University, Monterey Bay, Seaside, CA USA
aunfried@csumb.edu

Douglas Whitaker
Department of Mathematics and Statistics, Mount Saint Vincent University, Halifax, NS, Canada
douglas.whitaker@MSVU.CA

Marjorie E. Bond
Department of Statistics, Pennsylvania State University, State College, PA USA
meb6971@psu.edu

Leyla Batakci
Department of Mathematical Science, Elizabeth Town College, Elizabethtown, PA USA
batakcil@etown.edu

ABSTRACT

Attitudes play an important role in students' academic achievement and retention, yet quality tools to measure them are not readily available in the new field of data science. Through funding from the National Science Foundation, we are developing a family of instruments that measure attitudes toward data science in the context of an introductory, college-level course. This poster will showcase preliminary results discussing pilot instruments to assess instructors' attitudes toward teaching data science and an inventory which captures classroom characteristics. These instruments, based on Expectancy-Value Theory, will enable data science education researchers to evaluate pedagogical innovations and measure instructional effectiveness relating to student attitudes. We invite instructors of data science courses to join in this discussion and to use these instruments for their own data science education research projects.

CCS CONCEPTS

• Human-centered computing - Visualization • Information systems - Data structures • Information systems - Information integration • Mathematics of computing - Statistical paradigms • Computing methodologies - Modeling and simulation • Computing methodologies - Machine learning

KEYWORDS

Data Science, Attitudes, Instrument Design

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635525

ACM Reference format:
April Kerby-Helm, Michael A. Posner, Alana Unfried, Douglas Whitaker, Marjorie E. Bond, and Leyla Batakci. 2024. A Family of Instruments to Measure Data Science Attitudes. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA. ACM, New York, NY, USA. 2 pages.* https://doi.org/10.1145/3626253.3635525

1 ATTITUDES MATTER

As a new field, there is a clear lack of research about student attitudes in data science. Educators are therefore constrained to assuming the relevance of disciplinary based education research from fields which are closely related to data science, namely computer science and statistics, or from the literature on attitudes toward STEM in general. Across STEM programs, student attitudes are linked with student achievement and future career choices (e.g., [9]). In computer science, Gurer, Cetin, and Top [5] found that attitudes toward computer programming are associated with achievement in and perceived learning of computer science. Robinson, Pérez-Quiñones, and Scales [8] found that African American school-aged girls tend to have low attitudes toward computer science, but that appropriate interventions can improve these attitudes. In statistics, Evans [4] and Budé et al. [1], found a significant correlation between negative attitudes and poor achievement in undergraduate introductory statistics courses. The report Connecting Research to Practice in a Culture of Assessment for Introductory College-level Statistics [7], which was approved by the American Statistical Association, states the importance of understanding attitudes and identifies outcomes that can be realized when students' attitudes are adequately measured. Specifically, 1) how attitudes contribute to success in learning, in either the short or long term; (2) how attitudes contribute to long term engagement; and (3) what are the important attitude constructs to measure about instructors, and how do these influence teaching practices and ultimately student outcomes.

2 EXPECTANCY-VALUE THEORY

The theoretical framework guiding the development of the student and instructor instruments Expectancy-Value Theory (EVT) [2, 3], a psychological theory of motivation that has been thoroughly researched and widely adopted by researchers in many disciplines since it was first proposed nearly 40 years ago [3, 10, 11]. In EVT, one's achievement-related choices and performance are affected by how much one values the task (Subjective Task Values [STVs]) and what one expects to happen (Expectancy) - all other psychological components of the model are mediated by one's STVs and/or Expectancy [10]. EVT was chosen as the framework for these instruments because STVs and Expectancy have been shown to predict choices and performance on tasks and have been used extensively with undergraduate students [11].

3 FAMILY OF DATA SCIENCE INSTRUMENTS

In response to the need for validated, reliable instruments assessing student attitudes toward data science, the authors were awarded a grant under the National Science Foundation's Improving Undergraduate STEM Education program (DUE-2103392). The goal of the grant is to develop six instruments assessing student and instructor attitudes toward data science and statistics, as well as inventories of the learning environment. We refer to this research as Motivational Attitudes towards Statistics and Data science Education Research (MASDER).

Two workshops were hosted that brought together faculty from fields including computer science, statistics, education, and political science who taught introductory data science as well as data scientists from industry who work with undergraduate students. We met for a total of four days to discuss what data science is (and is not), identify content in an introductory data science class, and write specific items for the student data science survey (Student Survey of Motivational Attitudes toward Data Science (S-SOMADS)); the first pilot had 87 items measuring eight constructs.

Instructor attitudes affect teaching and the propensity to pursue and engage in professional development, as well having a direct effect on the attitudes of their students [6]. Our instructor survey instrument (Instructor Survey of Motivational Attitudes toward Data Science (I-SOMADS)), with 80 items and 10 constructs, is designed to be used with the student instrument to answer broader questions about the relationship between instructor attitudes and student attitudes and achievement.

To complement the measurement of student and instructor attitudes toward data science, we are also developing a learning environment inventory. The Environment Survey of Motivational Attitudes toward Data Science (E-SOMADS) inventory will measure course characteristics, the learning environment, and enacted classroom behaviors. For example, in what type of institution was this course taught? Did the instructor use peer evaluation for in-class coding activities? This will serve as a valuable tool to be used by itself. When used in conjunction with the other instruments, it will identify differences in student attitudes related to these characteristics as well as enable comparisons and research across contexts.

4 PRELIMINARY RESULTS

Preliminary results from the first pilot (n=402 students) of the S-SOMADS instrument will be discussed and items from the first I- and E-SOMADS pilot instruments will be presented.

ACKNOWLEDGMENTS

The authors would like to thank the MASDER Research Team. Part of this work is supported by the National Science Foundation DUE-2103392.

REFERENCES

[1] Budé, L. et al. 2007. Students' achievements in a statistics course in relation to motivational aspects and study behaviour. *Statistics Education Research Journal*. 6, 1 (2007), 5–21.

[2] Eccles[Parsons], J.S. et al. 1983. Expectations, values, and academic behaviours. *Achievement and achievement motives: Psychological and sociological approaches*. 76–146.

[3] Eccles, J.S. and Wigfield, A. 2020. From expectancy-value theory to situated expectancy-value theory: A developmental, social cognitive, and sociocultural perspective on motivation. *Contemporary Educational Psychology*. 61, (2020). DOI: https://doi.org/10.1016/j.cedpsych.2020.101859.

[4] Evans, B.R. 2005. *STUDENT ATTITUDES, CONCEPTIONS, AND ACHIEVEMENT IN INTRODUCTORY UNDERGRADUATE COLLEGE STATISTICS*. Temple University.

[5] Gurer, M.D. et al. 2019. Factors affecting students' attitudes toward computer programming. *Informatics in Education*. 18, 2 (2019), 281–296. DOI: https://doi.org/10.15388/infedu.2019.13.

[6] Martins, J.A. et al. 2012. Looking back over their shoulders: a qualitative analysis of portuguese teachers' attitudes towards statistics. *Statistics Education Research Journal*. 11, 2 (2012), 26–44.

[7] Pearl, D.K. et al. 2012. Connecting research to practice in a culture of assessment for introductory college-level statistics.

[8] Robinson, A. et al. 2016. African-American Middle School Girls: Influences on Attitudes toward Computer Science. *Computing in Science & Engineering*. 18, 3 (2016), 14–23. DOI: https://doi.org/10.1109/MCSE.2016.43.

[9] Schunk, D.H. 1991. Self-efficacy and academic motivation. *Educational Psychologist*. 26, 3–4 (1991), 207–231. DOI: https://doi.org/10.1080/00461520.1991.9653133.

[10] Wigfield, A. and Eccles, J. 2000. Expectancy–value theory of achievement motivation. *Contemporary Educational Psychology*. 25, 1 (2000), 68–81. DOI: https://doi.org/10.1006/ceps.1999.1015.

[11] Wigfield, A. and Eccles, J.S. 2019. 35 years of research on students' subjective task values and motivation: A look back and a look forward.

An Experience Report on Teaching a Large Introductory Programming Course in Hybrid Mode

Shyama Sastha Krishnamoorthy Srinivasan
IIIT-Delhi
Delhi, India
shyamas@iiitd.ac.in

Avi Gupta
IIIT-Delhi
Delhi, India
avig@iiitd.ac.in

Pankaj Jalote*
IIIT-Delhi
Delhi, India
jalote@iiitd.ac.in

ABSTRACT

As COVID made both students and teachers familiar with online teaching, it seems appropriate to explore a hybrid approach to teaching, in which the lectures are delivered in class but are simultaneously streamed online. An introductory programming course for an incoming cohort is frequently taught by having multiple sections, each being taught by an instructor, as colleges often have modest-sized classrooms that cannot accommodate the entire incoming cohort. Hybrid teaching can reduce the need for multiple instructors to teach different sections by having one large class. It also provides uniformity in instruction, which is hard to achieve with multiple sections. This poster shares our experience teaching a large introductory programming course in hybrid mode. The experience indicates that while in-class and overall attendance decline over the semester, online attendance is relatively steady and that most students attended a good fraction of lectures online. The data from in-class active learning quizzes also suggests that learning in lectures is similar for online and in-class students. Student feedback indicates that while a large fraction of students prefer in-class lectures for learning, a significant portion feels that both in-class and online are similar, and a fraction of students prefer online. Students' performance was somewhat correlated with total attendance but not the fraction attended in class or online. Overall, hybrid teaching of the introductory programming course has the potential advantage of increasing attendance and performance.

KEYWORDS

Hybrid Teaching, Hybrid Learning, Large Introductory Programming Course

ACM Reference Format:
Shyama Sastha Krishnamoorthy Srinivasan, Avi Gupta, and Pankaj Jalote. 2024. An Experience Report on Teaching a Large Introductory Programming Course in Hybrid Mode. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635486

*Advisor of this research work.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635486

1 INTRODUCTION

The Covid pandemic has ensured widespread use and familiarity with online teaching. Much of the online teaching during this period was done synchronously, where the lectures were given online in real time using Zoom, Google Meets, Webex, and more. Online teaching provides instructors the convenience of lecturing from any location and students of attending lectures remotely and having recorded lectures available later. At the same time, in-person has its advantages as an approach that has been refined over centuries in higher education. Post-COVID, it seems appropriate to explore a hybrid teaching approach in which perhaps the benefits of both can be combined.

In hybrid teaching, the lectures are delivered in class to students present but are also simultaneously streamed to online students. So, students can attend the session in class or online [5]. Moreover, Since the sessions are streamed using platforms like Zoom, Google Meets, Cisco, and more, they can be easily recorded and made available to all students. (It should be noted that blended education [2] is generally about using different forms of teaching for different portions of a course/module).

Hybrid teaching seems a natural extension of regular in-class teaching where computer-based projection methods (e.g., PowerPoint, writing on a tablet) are used to present to the class. It can also be viewed as a natural extension of synchronous online teaching where the online lecture is being given from the classroom.

Hybrid teaching can help reduce the number of instructors needed for teaching a large class. Often for teaching the introductory programming course, the incoming cohort is divided into sections, and each section is taught by a separate instructor, as no single classroom can accommodate the full incoming cohort. With hybrid teaching, the need for multiple sections is eliminated as all students are taught together in one large section, with some of the students joining the lecture in-class and others joining online. Having one instructor has a strong advantage of providing uniformity of instruction and assessment - something very challenging to achieve with multiple sections and instructors.

In this poster, we report our experience in teaching a large introductory programming course in a hybrid mode. The purpose of this experiment was to understand the effectiveness of Hybrid teaching. The same course was taught earlier in the in-class mode and during COVID, in an online mode. We share some analysis regarding the effectiveness of hybrid teaching and provide suggestions for its use in teaching introductory undergraduate courses.

2 TEACHING IN HYBRID MODE - OVERVIEW / METHODS / OBSERVATIONS

In this study, we organized the Introductory Programming course in a hybrid mode with a class strength of 636 students. Generally, these students will be divided into two sections with separate lectures by different faculty for each section. However, this semester, we experimented with having one combined section in which lectures will be given in hybrid mode. For video streaming, Zoom was adopted, and the Zoom screen was projected for in-class students. For audio streaming (as voice from in-class audio systems does not reach online students), a separate audio input device was chosen considering the clarity needed for the audience in the lecture hall without compromising the quality of audio in the Zoom call.

For labs and tutorials, the students were divided into 32 different groups, with Teaching Assistants (TAs) allotted to each group. This helped in managing the course effectively throughout the various assessments of the course. The various assessments included an initial quiz, labs, assignments, mid-sem, and final exams.

The lecturing approach utilized active learning approaches, which have been shown to promote learning during lectures through prior studies such as [3, 4]. For incorporating active learning, we used the locally developed mobile application ***ALT***, which provides for giving timed quizzes and compiling responses from the students. In each quiz, we incorporated peer-learning [1] to encourage intellectual discussions among the students by utilizing the think-pair-share approach. There were typically 3-4 quizzes in each class with a class duration of 1.2 hours. Students were given credit for their overall grade for responding to these quizzes.

The data from ALT quizzes, along with the Zoom attendance data, provided the vital measure needed to differentiate the attendance pattern of those who joined online from those in-class. We used the performance in the ALT quizzes to gauge the level of understanding in the lecture. The data suggests that the comprehension of lectures is similar for students attending online or in class. The feedback of students also indicated that most found in-class and online similar for learning. While a substantial fraction said that they prefer in-class, a reasonable fraction also indicated that they prefer online. This suggests that the hybrid mode of teaching can help all students - even those who prefer the online mode for their learning.

The data on attendance shows that while the in-class attendance in lectures declined as the semester progressed, as is generally observed, online attendance remained quite steady (around 160 students) throughout the semester. The data also shows that the fraction of lectures a student attends online is relatively independent of the total attendance of the student. This indicates that students generally prefer having the online option and use it frequently. It also suggests that overall attendance in lectures may improve in a hybrid mode. Increased attendance may improve learning - the experiment also showed that overall attendance is positively correlated (0.45) with overall grades. However, there was no significant correlation between the students' grades and the fraction of the classes attended either online (-0.11) or in-class (0.11). This seems to suggest that to improve student learning, a desired goal is to get them to attend lectures either in class or online - the mode of attendance is unimportant. The detailed plots explaining the trend among various metrics and measures are presented in the poster.

The main challenge in teaching such a large class in one section was managing and conducting various aspects of the course. For example, this course required a lab to be prepared each week and conducted by TAs in groups. It also required tutorial sheets to be prepared for weekly tutorials. These tightly scheduled activities posed the main challenge. TA management and coordination was another challenge, as there was a large number (40+) of TAs. In our course, we had two senior TAs who coordinated the TAs. Even with this, ensuring that TAs are responsive and do their tasks sincerely was a challenge, mainly since instructor oversight was not feasible.

All these students were also taking a few other courses in the semester - all of them in the traditional in-class mode. To gauge what their feeling is about teaching other introductory courses also in hybrid mode, we asked their view on this question for the other courses they do in their first semester (these are: Mathematics, Digital Circuits, Communication Skills, and Introduction to HCI). They were also given an option of "all" or "none ." Most of the students have shown a preference for all these courses to be taught in hybrid mode. This suggests that a hybrid teaching mode may be well suited for introductory courses where the matter to be taught is reasonably standard, and there is less need for discussions.

Though hybrid teaching was done in only one of the introductory courses, based on experience and student feedback, it seems that hybrid teaching may be a suitable approach for large introductory courses. This can help reduce the number of instructors needed and can also provide uniformity in instruction, which is hard to achieve if different sections are taught by different instructors. This teaching mode has the significant advantage of having recorded lectures available to students without requiring extra recording infrastructure. (In another feedback on what helped students learn well, "recorded lectures" were consistently mentioned as an important factor.) Hybrid teaching may also enhance class attendance – we know from experience that students miss lectures often, and attendance often falls dramatically towards the end of the semester; the online option may encourage students to attend more lectures, which can improve their learning.

Overall, we can say that hybrid teaching of a large introductory course has some advantages for students, as well as for institutions and faculty, and is well suited for teaching a large introductory class and may be better than entirely online or in-class teaching.

REFERENCES

[1] Catherine Crouch and Eric Mazur. 2001. Peer Instruction: Ten years of experience and results. *American Journal of Physics* 69 (08 2001). https://doi.org/10.1119/1.1374249ÍŤ

[2] Peter J Fadde and Phu Vu. 2014. Blended online learning: Benefits, challenges, and misconceptions. *Online learning: Common misconceptions, benefits and challenges* 9, 4 (2014), 33–48.

[3] Richard Felder and Rebecca Brent. 2009. Active learning: An introduction. *ASQ Higher Education Brief* 2 (01 2009).

[4] Michael Prince. 2004. Does Active Learning Work? A Review of the Research. *Journal of Engineering Education* 93 (07 2004), 223–231. https://doi.org/10.1002/j.2168-9830.2004.tb00809.x

[5] Annelies Raes, Pieter Vanneste, Marieke Pieters, Ine Windey, wim Van Den Noortgate, and Fien Depaepe. 2020-01-01. Learning and instruction in the hybrid virtual classroom: An investigation of students' engagement and the effect of quizzes.

Scaffolded Projects for the Social Good: A Strategy for Deploying Studio Model in CS Education

Stan Kurkovsky
kurkovsky@ccsu.edu
Central Connecticut State University
USA

Mikey Goldweber
mikeyg@denison.edu
Denison University
USA

Nathan Sommer
sommern1@xavier.edu
Xavier University
USA

Chad A. Williams
cwilliams@ccsu.edu
Central Connecticut State University
USA

ABSTRACT

Scaffolded Projects for the Social Good (SPSG) is an adaptable service-learning framework with a low adoption threshold based on the studio model. Its goal is to enable instructors to easily embed externally sourced projects supporting Computing for the Social Good (CSG) concepts into existing software engineering or similar courses and addresses the barriers common to service learning, as well as other frameworks with similar CSG-related objectives. Establishing connections between computing and its societal benefits has proven to be an effective strategy for attracting students, especially those from underrepresented groups within the discipline. Furthermore, this work supports competency-based learning by offering students an opportunity to solve real-world problems in an authentic environment using current industrial practices and tools coupled with strong mentoring support from volunteer professionals and near-peers. Using a studio model helps overcome the timing impedance between the length of a single academic term and the timeframes required to complete real-world projects with student teams.

ACM Reference Format:
Stan Kurkovsky, Mikey Goldweber, Nathan Sommer, and Chad A. Williams. 2024. Scaffolded Projects for the Social Good: A Strategy for Deploying Studio Model in CS Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635487

1 THE SPSG FRAMEWORK

Rationale: The concept of a software studio is based on the idea that a CS degree does not prepare a graduate to be a great software engineer any more than a degree in Arts prepares one to be a great artist. However, repeated practice under the guidance of experienced mentors within a studio framework can help develop student competencies to a significantly higher level [1]. The concept of a software studio has been successfully applied at both graduate [3, 4] and undergraduate levels [5, 6]. Current literature suggests that participating in software studio projects offers multiple advantages to students. These include fostering a heightened sense of ownership, which arises from the extended project duration and real-world application. Student learning is improved through mentoring by professionals, and students become more adept at using professional practices and tools. Additionally, these projects improve project outcomes through customer involvement and support reflective practices through retrospectives and peer feedback.

Implementing a studio-based approach to software projects is challenging. Setting up a logistical infrastructure requires a significant investment of faculty effort to create a project workflow that would fit their course or program needs. One of the primary challenges is that students participate in the studio projects for fixed periods of time (1, 2, or even 3 terms), while projects exist in the studio as long as needed. While some large institutions can provide staff and other resources to support such projects, this may be unrealistic for many teaching-focused institutions. There must be an established procedure for evaluating the project scope, which often relies on an internalized process involving program faculty, staff, and well-established partners sponsoring the projects. Each institution's studio approach is tightly integrated with the program curriculum and the institutional context, which could make this experience very difficult to adopt elsewhere.

The authors have accumulated a broad spectrum of experience working with various studio-based approaches for a wide variety of partners in over 75 projects. This experience has illuminated the challenges involved in such projects such as the difficulties of scaling the approach to large class sizes, identifying projects of an appropriate size, addressing project maintenance, and coordinating project handoff. Over time, these projects helped form a highly scaffolded framework that provided enough structure to ensure project success while also being flexible, so teams had sufficient agility for adapting to evolving requirements or emerging challenges.

Work Done: The authors are currently developing an easy-to-adopt framework to deploy studio-based projects, which is based on agile principles and the lessons learned from the HFOSS initiative [2]. It will provide instructors with a customizable set of curricular materials aimed to streamline the software project workflow and offer ample mechanisms for formative and summative assessment of student learning. Students participating in SPSG projects will gain

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635487

the experience of applying their knowledge and skills in an authentic project environment, preparing them to tackle the challenges of real-world projects. SPSG will help students make meaningful contributions to the missions of non-profit and community organizations while providing concrete examples regarding the impact of computing on society. SPSG will also make it easier for instructors to engage with external software project partners by providing a project feasibility assessment model.

Our current implementation of the SPSG framework builds on top of one or two upper-level courses commonly found in many programs: Software Engineering (SE) and Senior Project (SP). The SE course typically combines software engineering fundamentals with a semester-long course project. The SP course focuses primarily on the project and is often supplemented by readings and discussions emphasizing professional and ethical issues. A typical SPSG studio project can last 2 to 3 semesters. Students in both SE and SP courses can form teams working on a variety of projects for different customers. After the first semester, students who worked on a given project in SE, continue into SP and work with new SE student teams ensuring project continuity and knowledge transfer. Other options for project continuity are described below.

The Framework: Our framework uses a studio-based project organization that is grounded in agile practices by focusing on short iterations, partner (customer) involvement, and providing enough flexibility to address changing requirements and other challenges. Each course project consists of four phases: inception, elaboration, development, and transition. The inception phase occurs before the project commences and involves several key elements. During this phase, the instructor collaborates with the project partner to assess project feasibility, align it with the capabilities of student teams, define project scope and duration, and establish expected outcomes. This results in a standardized project proposal document, which is then received by one or more student teams.

During elaboration, students (or the instructor) form teams and work closely with the project sponsor to understand and formulate project requirements. For new projects, teams go through several iterations of project partner interviews to form outline requirements, which then are translated into user stories forming the initial product backlog. For continuing projects, remaining user stories are reviewed and adjusted taking into account any new user stories reflecting the next set of requirements. Teams work closely with the project partner to verify that the user stories correctly reflect the desired functionality, while the partner sets user story priorities.

The development phase is structured using the agile methodology. Each two-week sprint starts with the team planning their work by identifying which user stories will be moved from the product backlog into the sprint backlog. Given that a student might dedicate no more than 10 hours per week to the project, instead of daily scrum meetings, students participate in a weekly scrum during one of the regularly scheduled class meetings. During a weekly scrum, each team briefly discusses their progress made during the past week towards meeting the sprint goal, as well as the plan for the upcoming week. Throughout the sprint, teams maintain an open line of communication with the project partner to make sure that any questions can be answered promptly. At the end of the sprint, each team demonstrates their progress to their project partner during a sprint review when they may accept or reject some or all of the newly developed functionality. Each team also participates in an in-class self-reflection during the sprint retrospectives when they can inspect the results they accomplished, discuss the lessons learned during the sprint, and adapt both the product backlog (to better reflect any possible changes in the project requirements) and the team's workflow (to make sure that the team does more of what works for them and less of what doesn't).

The transition phase takes place during the last week of the semester. It focuses on knowledge transfer and typically includes the final project demonstration to the project partner and the entire class. There are several possibilities at this point: a) the project continues next semester with the same team(s) (typical for teams in the SE course that will be in the SP course next semester); b) the project continues next semester with at least one new team (same scenario as above, but with an additional team from the next semester's SE course; or when there's no overlap in teams); or c) the project is completed and there are no teams actively involved in it (aside from routine maintenance).

Regardless of the project state, to facilitate the knowledge transfer, teams typically prepare a set of user and deployment documents for the project partner and the team(s) that may continue working on the same project. Project continuity is enabled by several factors: a) team staggering so that there is an overlap between a team continuing with the project and a new team, b) extensive knowledge transfer documentation including configuration and build documents, and c) near-peer mentoring where a recent graduate who worked on the same project mentors the next team(s).

Throughout their work on the projects, student teams are typically required to produce a range of deliverables in the form of reports, presentations, source code commits, project demonstrations, meeting notes, etc. Each of these deliverables is an opportunity for formative assessment, while those produced towards the end of the project and especially at the project transition phase are very useful for summative assessment of student learning.

2 ACKNOWLEDGEMENTS

This work was supported in part by NSF awards 2315322 and 2315323.

REFERENCES

[1] Christopher N. Bull and Jon Whittle. 2014. Supporting Reflective Practice in Software Engineering Education through a Studio-Based Approach. *IEEE Software* 31, 4 (2014), 44–50. https://doi.org/10.1109/MS.2014.52

[2] Heidi J. C. Ellis, Gregory W. Hislop, Stoney Jackson, and Lori Postner. 2015. Team Project Experiences in Humanitarian Free and Open Source Software (HFOSS). *ACM Trans. Comput. Educ.* 15, 4, Article 18 (dec 2015), 23 pages. https://doi.org/10.1145/2684812

[3] Phillip A. Laplante. 2006. An Agile, Graduate, Software Studio Course. *IEEE Transactions on Education* 49, 4 (2006), 417–419. https://doi.org/10.1109/TE.2006.879790

[4] David Root, Mel Rosso-Llopart, and Gil Taran. 2008. Exporting Studio: Critical Issues to Successfully Adopt the Software Studio Concept. In *2008 21st Conference on Software Engineering Education and Training*. 41–48. https://doi.org/10.1109/CSEET.2008.21

[5] Daniela Rosca. 2018. Acquiring Professional Software Engineering Skills through Studio-based Learning. In *2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*. 1–6. https://doi.org/10.1109/ITHET.2018.8424773

[6] Robbie Simpson and Tim Storer. 2017. Experimenting with Realism in Software Engineering Team Projects: An Experience Report. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. 87–96. https://doi.org/10.1109/CSEET.2017.23

Poprawa: A Gradebook Library for Alternatively-Graded Courses

Zachary Kurmas
kurmasz@gvsu.edu
Grand Valley State University
Allendale, MI, USA

ABSTRACT

We present `Poprawa`, a library to help instructors who use alternative grading techniques prepare and securely disseminate progress reports to their students.

Alternative grading techniques such as mastery grading, specifications grading, and "ungrading" are becoming increasingly popular. However, they are not well-supported by many learning management systems (LMS). Given the current state of LMS gradebooks, many instructors prefer to track student progress using a spreadsheet. Doing so presents two challenges: (1) It can be tedious and time-consuming to generate progress reports that reflect the instructor's unique grading criteria, and (2) it can be difficult to securely provide students with electronic access to those reports.

The `Poprawa` library addresses both issues. (1) It provides a simple means of accessing data in an Excel spreadsheet so that instructors can quickly and easily write code to construct progress reports customized for the course; and (2) it utilizes `git` and GitHub Classroom to provide a workflow that automatically uploads progress reports to a each student's private repository where he or she can conveniently and securely view it. `Poprawa` also provides hooks to allow instructors to disseminate progress reports without using `git` and/or GitHub.

ACM Reference Format:
Zachary Kurmas. 2024. Poprawa: A Gradebook Library for Alternatively-Graded Courses. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635488

1 INTRODUCTION

Alternative grading techniques such as mastery grading, specifications grading, and "ungrading" are becoming increasingly popular [2, 3, 5–7]. However, one challenge in adopting these techniques is that, in general, they are not well-supported by many current learning management systems (LMS) gradebooks. For example, several of the more popular LMSs make it awkward to record non-numeric feedback; worse yet, few LMSs provide sufficient flexibility in their gradebooks to support the algorithms instructors prefer to use when summarizing overall progress of assigning final letter grades.

Solutions to this mismatch between grading practices and gradebook features include combinations of (1) preparing progress reports by hand, and distributing them in class (or emailing them individually); (2) "encoding" student progress in a format supported (or preferred) by the LMS, then teaching students to interpret the "encoded" data [1, 4], and (3) modifying the course's grading system to conform to the options provided by the gradebook.

Our contribution is to introduce `Poprawa`, a library that helps instructors who use alternative grading practices by providing a tool to help them track and communicate student progress in a manner that is convenient, secure, and does not require the instructor make the grading criteria "feel" numeric (or otherwise modify his or her practice to conform to the LMS).

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635488

2 POPRAWA

`Poprawa` is a Ruby library. It also comes with a script that uses the library's three main features. Running this script (1) retrieves student progress data from an Excel spreadsheet, (2) generates a progress report for each student, and (3) places that report in the student's `git` repository then pushes the changes so that the student can securely view the report by accessing his or her GitHub account. (See Section 2.3 for details.)

The script allows instructors to use `Poprawa` without modification if their system of assigning grades is sufficiently similar to ours. However, because alternative grading practices differ significantly among instructors (sometimes even among courses taught by the same instructor), we designed `Poprawa` be more of a library than an "off-the-shelf" product.

Our target audience is instructors in the field of computing, so we focused on a solution that allows users to customize the tool by writing code, rather than trying to make everything configurable. Trying to generalize all the innovative alternative grading practices we've learned about seems like a fool's errand. Instead, our goal is to find the right balance between configurability (setting properties in a file) and customizability (writing small sections of code). This balance is critical, because requiring too much of either can make it so that it is easier for an instructor to write his or her own tool than to learn how to use ours.

As a library, `Poprawa` has three main components:

2.1 Excel Spreadsheet Interface

`Poprawa` provides a simple means of retrieving the marks stored in an Excel workbook as a "three-dimensional hash" (assignment type, assignment name, and student). `Poprawa` uses the `rubyxl` gem to access the spreadsheet data; but, hides the complexity of navigating the various data structures that represent formatting, formulas, and other details of an Excel workbook.

Our tool also allows instructors to optionally store the mark, the number of "late tokens" used, and/or an additional comment all in the same spreadsheet cell. (The fields are delineated by a pipe and a semi-colon respectively. `Poprawa` separates these fields when parsing the spreadsheet.) We provide the ability to place all three

values in the same cell because, in our experience, relatively few assignments require either a late token or a comment. Creating a separate, but sparsely populated column for each assignment resulted in a lot of "visual clutter". We also found Excel's comment feature cumbersome to use. Instructors who make heavier use of late tokens or comments (e.g., tend to have a comment for each assignment) can create separate columns if they choose.

2.2 Report Generation

`Poprawa` generates a report in Markdown format for each student that lists (a) the marks for each assignment, and (b) the progress toward a given final letter grade for the course.

In our experience, the algorithm for calculating a student's final grade differs significantly among instructors who use alternative grading practices. Therefore, we made a point of isolating the methods that implement this algorithm so that instructors can easily provide their own implementation. However, because the vast majority of instructors appear to use an algorithm that is primarily based on the number of marks that indicate "success" or "meets expectations", we don't expect that our code will require much change in most cases.

2.3 Dissemination

We disseminate reports by placing them in the `README.md` file in a `git` repository, then pushing that repository. Using the `README.md` file allows students to view the report by simply visiting the repository's main web page. Students need not clone or pull the repository (hence the use of Markdown).

We use GitHub Classroom to generate a repository for each student. We configure Classroom to create private repositories; therefore, the confidentiality of the student's data is protected by GitHub's security. (This process does require that we manually collect each student's GitHub account name and enter it in the spreadsheet. Our class sizes average 30; so this is not cumbersome.)

It would not be difficult to write code to disseminate reports differently (email, upload to an LMS, print and hand out, etc.).

3 RELEVANCE AND PURPOSE

`Poprawa` serves two main purposes: (1) It provides an immediate solution for instructors who are not happy with the gradebook options their LMS currently provides, and (2) it provides a "sandbox" that the community can use to try out different approaches to reporting progress and calculating grades before making specific requests of LMS providers.

We are concerned that instructors who resign themselves to working within the limited framework provided by their LMS may unintentionally get in the habit of viewing the problem through the "lens" shaped by those constraints and, therefore, fail to imagine more innovative options. In contrast, providing instructors a library-based "sandbox" with few limitations will hopefully contribute to the creation of more flexible and innovative designs.

We have two main purposes in presenting `Poprawa` as a SIGCSE poster: First, we want to solicit feedback from the community to see what features are desired and how we could make the tool easier to use. `Poprawa` is designed for instructors who have strong coding skills. The learning curve for other instructors will be steep. We hope to receive feedback that will help us find ways to make this tool available to users with less coding experience (including users outside computing-related fields) while also maintaining its ease of use. We also would like to learn how instructors who use GitHub in large couses link GitHub usernames to students in a reliable and efficient manner.

Second, `Poprawa` has the potential to be a platform with which we can research how to most effectively answer the questions "How am I doing in this couse?" or "What letter grade am I on track to get?" It is often not obvious how best to answer this question in alternatively graded courses because, when applying the guidelines as written, a student often technically has a very low grade until late in the semester when he or she has completed most of the course's learning targets. Attempting to pro-rate the requirements for a given letter grade based on the percentage of the semester completed can be problematic: When the number of learning target is small, the resulting rounding error can be misleading. (For example, one time an"A" student got a "retry" on one quiz and the projected grade became a "C".)

Many instructors present student progress as a set of "boxes" that are marked when the corresponding requirement is complete. In classes with several categories of requirements, it can be difficult to determine at a glance what final grade the student's curent trajectory is heading toward.

We are currently prototyping visualizations of overall progress and look forward to discussing other ideas and/or potential collaborations with the audience.

This is our third semester using `Poprawa`, and we have found that it significantly simplifies the process of tracking student progress as compared to the gradebook offered by our university's LMS.

4 AVAILABILITY

Poprawa is available on GitHub:

https://github.com/kurmasz/poprawa/

REFERENCES

[1] 2023. *Integration with LMS.* https://sbghub.lmu.build/implementation/integration-with-lms/

[2] Andrew Berns. 2020. Scored out of 10: Experiences with Binary Grading Across the Curriculum. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 1152–1157. https://doi.org/10.1145/3328778.3366956

[3] L. Nilson and C.J. Stanny. 2015. *Specifications Grading: Restoring Rigor, Motivating Students, and Saving Faculty Time.* Stylus Publishing. https://books.google.com/books?id=UrCpCwAAQBAJ

[4] Abigail Noyce and David Largent. 2023. *Using the Canvas Gradebook with Specifications Grading.* https://gradingforgrowth.com/p/using-the-canvas-gradebook-with-specifications

[5] Claudia Ott, Brendan McCane, and Nick Meek. 2021. Mastery Learning in CS1 - An Invitation to Procrastinate?: Reflecting on Six Years of Mastery Learning. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1* (Virtual Event, Germany) *(ITiCSE '21)*. Association for Computing Machinery, New York, NY, USA, 18–24. https://doi.org/10.1145/3430665.3456321

[6] Ellen Spertus and Zachary Kurmas. 2021. Mastery-Based Learning in Undergraduate Computer Architecture. In *ACM/IEEE Workshop on Computer Architecture Education, WCAE 2021, Raleigh, NC, USA, June 17, 2021*. IEEE, 1–7. https://doi.org/10.1109/WCAE53984.2021.9707147

[7] Ella Tuson and Timothy Hickey. 2023. Mastery Learning with Specs Grading for Programming Courses. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1049–1054. https://doi.org/10.1145/3545945.3569853

Exploring AIFORGOOD Summer Camp Curriculum to Foster Middle School Students' Understanding of Artificial Intelligence

Kyungbin Kwon
Indiana University
Bloomington, U.S.A.
kwonkyu@indiana.edu

Keunjae Kim
Indiana University
Bloomington, U.S.A.
kimkeun@iu.edu

Anne T. Ottenbreit-Leftwich
Indiana University
Bloomington, U.S.A.
aleftwic@indiana.edu

Krista D. Glazewski
North Carolina State University
Raleigh, U.S.A.
kdglazew@ncsu.edu

Matthew L. Brown
Indiana University
Bloomington, U.S.A.
mb2@iu.edu

Haesol Bae
University at Albany, State University of New York, U.S.A.
hbae4@albany.edu

Florentina M. Closser
Indiana University
Bloomington, U.S.A.
fclosser@iu.edu

ABSTRACT

This study explores the structure of a summer camp curriculum and the development of middle school students' understanding of Artificial Intelligence (AI). Pre-posttests and surveys measuring AI knowledge and attitudes toward AI were analyzed by paired t-tests. Thematic analysis was used to analyze over 30 hours of video footage and student artifacts to examine the integration of curriculum and the development of students' understanding. The curriculum encompassed machine learning (ML), natural language process (NLP), and computer vision (CV), focusing on improving students' understanding of AI through the creation of AI-based artifacts, and mini-projects addressing daily issues faced in homes and schools. The findings revealed that students' overall AI knowledge and attitudes toward AI significantly improved after the intervention. Additionally, students recognized the importance of data quality and quantity in training AI. These findings highlight the positive impact of summer camp curriculum with hands-on experiences, beneficial for fostering middle school students' understanding of AI.

CCS CONCEPTS

• Security and privacy ~ Social aspects of security and privacy • Computing methodologies ~ Artificial intelligence • Applied computing ~ Interactive learning environments • Social and professional topics ~ Computing literacy • Social and professional topics ~ K-12 education

KEYWORDS

AI education, summer camp curriculum, middle school

ACM Reference format:
Kyungbin Kwon, Keunjae Kim, Anne T. Ottenbreit-Leftwich, Krista D. Glazewski, Matthew L. Brown, Haesol Bae, and Florentina M. Closser. 2024. Exploring AIFORGOOD Summer Camp Curriculum to Foster Middle School Students' Understanding of Artificial Intelligence. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635536

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635536

1 BACKGROUND OF THE STUDY

Artificial Intelligence (AI) has become increasingly integrated into various technologies, permeating our daily lives rapidly. Following this, teaching AI has gained significant attention in K-12 AI education, with a focus on cultivating necessary future skills such as problem-solving and creativity (Sanusi et al., 2023). By understanding AI concepts and principles, students can actively engage with AI technologies, becoming creators and designers rather than passive consumers with nurturing future skills [1]. Previous studies have emphasized the design and implementation of AI-focused curricula, targeted middle school students, and promoting teaching and learning about AI.

Despite these endeavors, room for enhancement remains in teaching and learning of AI in K-12 contexts, primarily due to the lack of AI curricula tailored to age-appropriate activities. Given the intricate and challenging nature of AI concepts and principles relative to the comprehension level of middle school students, the task of devising effective learning activities to convey clear AI ideas

becomes challenging. Prior research has already emphasized the limitations of AI curricular approaches that either offer a limited choice of hands-on experiences or lack incorporating authentic contexts and real-world issue-solving [2]. These approaches could constrain students' autonomy as learners and hinder their active engagement with AI technologies. In this regard, this study aims to design the AI curriculum and explore the students' understanding of AI through hands-on practices. The study addresses the following research questions: RQ1: What is the summer camp curriculum that supports students' understanding of AI? RQ2: How does the understanding of AI develop among students?

2 METHOD

2.1 Research Design

This study employs a case study design to explore the development of students' comprehension of ML within the AI FOR GOOD summer camp, introducing the AI concepts and principles to middle school students from rural areas. By designing the hands-on practices and discussions to support students' understanding of AI and observing the students' practical exercises and discussions, the study aimed to broaden our understanding of how concepts and principles related to AI develop within the specific context of the camp.

2.2 Participants and Contexts

The researcher recruited participants from public middle schools in the Midwestern region of the United States. Consent was obtained from the participants and their parents, comprising a total of 17 students with 12 boys and 5 girls from 6th to 9th grades. The researchers developed and implemented a curriculum for the AI FOR GOOD summer camp, which spanned five days, encompassing ML, NLP, and CV concepts.

2.3 Data Collection and Analysis

Data was collected and analyzed through two phases. First, pre-posttests were conducted to investigate the development of students' understanding of AI. Those measured knowledge of AI concepts and surveys for measuring attitudes toward AI. A paired t-test was conducted to compare the means from before and after the intervention. Next, video data over 30 hours were recorded capturing summer camp activities and interactions. Additionally, students' learning artifacts like mini-project proposals were collected to achieve an in-depth understanding of the development of students' understanding of ML. The collected artifacts, videos, and interview transcripts were analyzed through the inductive thematic analysis method [3].

3 PRELIMINARY FINDINGS AND FUTURE STUDIES

3.1 Designed Summer Camp Curriculum

The curriculum aimed to promote students' understanding of AI concepts and principles and their practical application to solve daily problems. It spanned five days curriculum, encompassed ML, NLP, and CV, with activities from day 1 to day 4, including unplugged activities to grasp the main ideas of AI, training and programming AI models with computers, creating AI-based artifacts with AI-based tools, and designing prototypes with mini-projects thinking about solving problems in daily basis. For day 5, students planned the AI-based solution to solve the problems on a daily basis by employing the knowledge and skills they learned from the previous sessions.

3.2 Students' understanding of AI

A paired t-test was conducted to assess the impact of the curriculum on students' knowledge of AI concepts by comparing the scores. The results indicated that there was a significant difference between the pre-test (M = 11.12, SD = 3.18) and post-test (M = 13.00, SD = 2.03); t (16) = 2.52, p <.001. Likewise, a paired t-test was performed to investigate how the curriculum impacted on students' attitudes toward AI. The results indicated that there was a significant difference between the pre-test (M = 3.80, SD = .62) and post-test (M = 4.18, SD = .53); t (16) = 3.60, p <.001. The overall results indicated that the summer camp curriculum had a positive impact on students' comprehensive understanding of AI as well as attitudes toward AI.

Moreover, students showed an improved understanding of AI, particularly recognizing the significance of the quality and quantity of training data. During the creation of AI-based artifacts, students gained a deeper understanding of the crucial roles of data quality and quantity. As they iteratively modified their artifacts and AI models, they realized they often didn't meet their initial expectations. For instance, when creating an AI squat machine using pose recognition, most students recognized that the more data they put into each label with balanced numbers, the better outcomes from AI. Also, some students took steps to improve data quality. Some students tried to capture pictures of squat poses from different angles or capture the background image to provide more representative data and reduce the noise to enhance the quality of data for ML algorithms.

ACKNOWLEDGMENTS

This material is based upon work supported by the Department of Defense National Defense Education Program under Grant No. W15QKN-20-1-1000.

REFERENCES

[1] Dipaola, D., Payne, B. H., & Breazeal, C. (2022). Preparing children to be consumers and designers of AI technologies. In Kong, S. C., & Abelson, H. (Eds.). (2022) *Computational thinking education in K-12: Artificial intelligence literacy and physical computing* (pp. 182–205). MIT Press.

[2] Carney, M., Webster, B., Alvarado, I., Phillips, K., Howell, N., Griffith, J., Jongejan, J., Pitaru, A., & Chen, A. (2020). Teachable machine: Approachable Web-based tool for exploring machine learning classification. In *Proceedings of Extended abstracts of the 2020 CHI conference on human factors in computing systems* (pp. 1-8). ACM. https://doi.org/10.1145/3334480.3382839

[3] Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2), 77-101. https://doi.org/10.1191/1478088706qp063

Facilitating Teens as Ethical Sensemakers of Technology

Rotem Landesman
University of Washington
Seattle, Washington, USA
roteml@uw.edu

Jean Salac
University of Washington
Seattle, WA, USA
salac@uw.edu

Amy J. Ko
University of Washington
Seattle, WA, USA
ajko@uw.edu

ABSTRACT

With the growing awareness of emerging technologies' impacts on teens' lives, families, and communities, rethinking the ways in which we educate and talk about these innovations and their moral and ethical complexities are gaining steam. We present a novel pedagogical intervention that blends techniques from *Philosophy for Children (P4C)*, a pedagogical approach that teaches youth reasoning and argumentative skills, with *Youth as Philosophers of Technology*, a framework for computing education that foregrounds learning how to decode and unmake tech's relationship with power through artistic, moral and humanistic inquiry, without devaluing core computing practices, such as design, making, coding, and tinkering. We studied this intervention in a summer elective class with 12 students ages 14-18 in the US. Our ongoing data analysis revealed two categories of themes: (1) 'launchpads for ethical sensemaking', namely instances *when* we observed ethical sensemaking around technology, and (2) 'expressions of ethical sensemaking', namely *what* students' ethical sensemaking looked like when discussing the ethical implications of technology. We hope to catalyze discussions for both researchers on characterizations of and growth around ethical sensemaking of technology, as well as practitioners on implementations of *Youth as Philosophers of Technology* and *P4C* ideas in their classrooms.

ACM Reference Format:
Rotem Landesman, Jean Salac, and Amy J. Ko. 2024. Facilitating Teens as Ethical Sensemakers of Technology. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635510

1 MOTIVATION & BACKGROUND

From the controversies around large language models in schools [5] to social media algorithms encouraging compulsive technology use [1], emerging technologies have indelibly shaped the lives, families, and communities of today's teenagers. Scholars in computing education research (CER) have thus explored ways to educate teens that prioritizes the questioning of the moral and ethical implications of technology, also known as critical computing education [2, 3]. Critical approaches to computing education have largely comprised of 'critical inquiry', which centers the analysis of power dynamics and suggestions of possibilities for change, to 'critical design', which centers the (re)design with computing in ways that aspire towards justice and change, to 'critical reimagination', which centers rethinking the present and the past to critically reimagine computing and technology for more equitable and just futures [6].

One approach to critical inquiry positions youth as 'philosophers of technology' [9], as active thinkers of the role of technology in their lives. In this approach, core computing practices are decentered without being devalued. Instead, what is pedagogically prioritized is youth learning to wrestle with the multiplicities, inconsistencies, and ethical implications of technology in local and global contexts. Consistent with this approach, the field of philosophy for children (P4C) offers techniques to support youth in this type of learning. P4C [4, 8] emerged as an approach to engage children in representing, discussing, and working through the fundamentally philosophical questions that they often encounter as they go through the world. P4C requires developing both critical thinking and good discourse skills — being able to think beyond simply recalling the facts, reflecting on one's own positionality and listening to a peer speak and respond thoughtfully. These are all skills that are crucial to develop if youth are to be active thinkers of technology's effects on their lives, families, and communities.

2 METHODS

To investigate how students may grow as ethical sensemakers of technology, we developed a novel pedagogical intervention which combined *Youth as Philosophers of Technology* [9] with techniques from *P4C* [4]. In our intervention, we drew from the basic principles of Vakil & McKinney de Royston's framework: (1) centering the relationality of the subject matter to others, to place, and to sociopolitical realities and histories, (2) highlighting analyses of both the tech stack and the cultural, social, and political contexts in which technology is used, and (3) encouraging design as a way to express ethical sensemaking. Simultaneously, we also leveraged some P4C techniques, namely creating a community of inquiry and adapting tools from philosophy as age-appropriate scaffolds. We specifically drew from a tool called *Moral Prisms*, which offers age-appropriate representations of common Western moral theories [7]:

- The Existentialist prism asks: "What course(s) of action will set people most free?"
- The Deontological prism asks: "What would I do if everyone in the world were to do as I did?"
- The Ethic of Caring prism asks: "What course(s) of action will best sustain and nurture a caring relationship between myself and others?"
- The Communitarian prism asks: "How would I act if everyone in my community knew exactly what I were doing?"
- The Utilitarian prism asks: "What course(s) of action will best maximize total happiness in the world?"

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635510

Week (Topic)	Activities
Week 1 (Intro)	Formation of the Community of Inquiry Intro to the Moral Prisms through debate around a local ban on cashless businesses in *Anon. City* Student selection of class topics
Week 2 (Data Privacy)	'What is my Digital Footprint?' Activity Analysis & Discussion of *Anon. State* Bill to Protect Children's Rights in Parent-Influencer Content
Week 3 (Social Media)	'Social Media Scavenger Hunt' Activity Analysis & Discussion of *Anon. City* Public Schools lawsuit against Social Media companies
Week 4 (AI)	'Train your own ML Model' and 'Break ChatGPT' Activity Analysis & Discussion of *Anon. City* Public Schools' ban on ChatGPT
Weeks 5-6	Final Project: Analysis & Design of an Alternative Future story on a student-selected contemporary moral dilemma

Table 1: Timeline and Overview of Class Activities

- The Virtue Ethics prism asks: "What would the most virtuous person I know of do in this situation?"
- The Egoist prism asks: "What course(s) of action will most effectively ensure that my short- and long-term goals are reached?"

We implemented our intervention in an elective class within a 6-week summer program (June-August 2023) at a northwest United States university aimed at students ages 14-18 from local under-resourced schools who were low-income and/or the first in their family to pursue a post-secondary education (i.e. first-generation). After we described the risks of research participation (namely discomfort from discussing potentially negative experiences with technology), all 12 students enrolled in our class assented to their classwork being analyzed for research through a form administered on the first day of class. In the first week of class, students engaged in rapport-building activities to develop their community of inquiry and collaboratively selected the three topics covered in the class. To introduce them to the Moral Prisms tool presented above, we discussed a social dilemma centering around technology use, and then showcased how a possible solution to the dilemma could be altered by choosing to look at it through one, or several, of the moral prisms. In Weeks 2-4, we covered each of the student-selected topics, adhering to the following structure. We started each week with an activity to introduce them to the topic, then in groups of 3-4, students analyzed a local contemporary issue related to the topic through a subset of the moral prisms, and finally, students presented their analysis to the class to spark a class-wide discussion. In Weeks 5-6, students worked on their final projects where in groups of 3-4, they selected a contemporary moral dilemma to analyze and designed an alternative future story based on that dilemma. An overview of class activities is shown in Table 1. Data analysis included a mixed inductive and deductive thematic approach, with students' classwork and daily instructor reflections as our main data source.

3 PRELIMINARY RESULTS

Our ongoing analysis has revealed two categories of themes. The first category is comprised of 'launchpads for ethical sensemaking', namely instances *when* we observed ethical sensemaking around technology. Students often made sense of the moral and ethical complexities around technology when they (1) made connections to their lives outside the classroom, (2) collaborated with their peers, and (3) engaged with the scaffolding provided by the moral prisms. The second category consists of students' 'expressions of ethical sensemaking', namely *what* students' ethical sensemaking looked like when discussing the ethical implications of technology. Students' often showed their sensemaking and made sense of such implications through (1) questioning the definitions of ideals, such as happiness and freedom, (2) rejecting the dichotomy of a binary good and bad with respect to technology, (3) wrestling with dissonance and contradictions, and (4) deciding whether or not to be flexible with their principles.

Through a SIGCSE 2024 poster presentation, we hope to spark discussions for both researchers and practitioners alike. Among researchers, we invite conversations around how we may characterize and evaluate ethical sensemaking around technology, and what student growth as philosophers of technology could look like in different contexts. As for practice, we would like to discuss how teachers and curriculum developers could incorporate principles of *Youth as Philosophers of Technology* and *P4C* in their classrooms.

ACKNOWLEDGMENTS

We would to thank Jared Ordoña-Lim for co-instructing the class, and Roseann London and Kristina Owsinski for organizing Upward Bound 2022 Summer Academy. This material is based upon work supported by the National Science Foundation under grants 2127309, 1539179, 1703304, 1836813, 2031265, 2100296, 2122950, 2137834, 2137312, and unrestricted gifts from Microsoft, Adobe, and Google.

REFERENCES

[1] Katie Davis, Petr Slovak, Rotem Landesman, Caroline Pitt, Abdullatif Ghajar, Jessica Lee Schleider, Saba Kawas, Andrea Guadalupe Perez Portillo, and Nicole S Kuhn. 2023. Supporting Teens' Intentional Social Media Use Through Interaction Design: An exploratory proof-of-concept study. In *Proceedings of the 22nd Annual ACM Interaction Design and Children Conference*. 322–334.

[2] Yasmin Kafai, Chris Proctor, and Debora Lui. 2020. From theory bias to theory dialogue: embracing cognitive, situated, and critical framings of computational thinking in K-12 CS education. *ACM Inroads* 11, 1 (2020), 44–53.

[3] Amy J Ko, Alannah Oleson, Neil Ryan, Yim Register, Benjamin Xie, Mina Tari, Matthew Davidson, Stefania Druga, and Dastyni Loksa. 2020. It is time for more critical CS education. *Commun. ACM* 63, 11 (2020), 31–33.

[4] Matthew Lipman. 1982. Philosophy for children. *Thinking: The Journal of Philosophy for Children* 3, 3/4 (1982), 35–44.

[5] Chung Kwan Lo. 2023. What is the impact of ChatGPT on education? A rapid review of the literature. *Education Sciences* 13, 4 (2023), 410.

[6] Luis Morales-Navarro and Yasmin B Kafai. 2023. Conceptualizing Approaches to Critical Computing Education: Inquiry, Design, and Reimagination. In *Past, Present and Future of Computing Education Research: A Global Perspective*. Springer, 521–538.

[7] PLATO. 2022. https://www.plato-philosophy.org/teachertoolkit/moral-spectrum-exploration-exercise/

[8] Steve Trickey and Keith J Topping*. 2004. 'Philosophy for children': A systematic review. *Research papers in Education* 19, 3 (2004), 365–380.

[9] Sepehr Vakil and Maxine McKinney de Royston. 2022. Youth as philosophers of technology. *Mind, Culture, and Activity* (2022), 1–20.

DCC Sidekick: Helping Novices Solve Programming Errors Through a Conversational Explanation Interface

Lorenzo Lee Solano
LorenzoLeeSolano@gmail.com
University of New South Wales
Sydney, NSW, Australia

Jake Renzella
jake.renzella@unsw.edu.au
University of New South Wales
Sydney, NSW, Australia

Alexandra Vassar
A.vassar@unsw.edu.au
University of New South Wales
Sydney, NSW, Australia

ABSTRACT

Students in introductory computing courses often lack the experience required to effectively identify and resolve errors in their code. For such students, Programming Error Messages (PEMs) are often the first indication of an error, and could provide valuable debugging guidance. However, in many cases, such as with standard C compiler implementations, PEMs are largely unsuitable for novices. Confusing, misleading, and filled with terse language and jargon, these messages instead act as an additional source of difficulty.

In this paper, we present *DCC Sidekick*, which integrates the Debugging C Compiler (DCC) with a Large Language Model (LLM) in a web-based dashboard to produce contextual, accurate guidance conducive to student learning. This dashboard is directly accessible from the output of the compiler, and provides a *bird's-eye-view* of the program source, compiler output, and a conversational AI interface to help unravel cryptic error messages. We aim to deploy DCC Sidekick to a C-based CS1 cohort at a large higher education institution to investigate how novice students utilise the conversational explanation interface during debugging activities. In this work, we present our experience designing and building DCC Sidekick.

ACM Reference Format:
Lorenzo Lee Solano, Jake Renzella, and Alexandra Vassar. 2024. DCC Sidekick: Helping Novices Solve Programming Errors Through a Conversational Explanation Interface. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635483

1 INTRODUCTION

Programming Error Messages (PEMs) are typically generated by the compiler or interpreter of the programming language. These messages act as a form of immediate feedback for programmers, and can be essential in guiding the process of debugging [12].

However, for novices in particular, these PEMs can be a significant source of confusion and stress [3, 11], and often act as a major barrier to progress in introductory programming courses [2]. Previous work shows that CS1 students struggle significantly with debugging, encountering errors even on simple activities [7], which they then often fail to identify and correct [17]. One of the key difficulties in mastering this skill is that debugging mostly happens outside of the classroom, meaning that students lack direct support whilst undertaking this task. Unfortunately, the notoriously cryptic nature of PEMs makes them at best, largely ineffective, and at worst, actively detrimental, in assisting novices with debugging [1, 9].

The C language, while still historically significant and industry relevant, has seen a decrease in popularity in a pedagogical context [15]. Numerous complaints have been made in regards to the novice debugging experience and error message quality of the language [6, 10]. Despite these challenges, some institutions stand by the use of C in CS1, to help foster a foundational understanding of computation and to prepare students for later courses [16]. However, more work must be done to design tools which adapt the language and improve its accessibility for novices.

Improving the efficacy of PEMs has proven to be difficult, with attempts hampered by both technical limitations and a lack of clear consensus on effective error message design [3]. As such, the approaches to error message enhancement have varied significantly in method, and have often found inconclusive results [14]. Recent progress has been made in error message readability and enhancement, demonstrating the potential for the practical improvement of PEMs [4, 8]. In addition, developments in Large Language Model (LLM) and Generative AI technology have presented new opportunities to address previous limitations of PEM enhancement [13].

In this work, we investigate the promising application of Generative AI to provide a conversational interface to explain error messages. Our tool, DCC Sidekick, aids with interpreting and enhancing PEMs to scaffold student understanding of debugging. The conversational format, which results from context-rich prompts generated at the compiler level, could help to provide more effective debugging guidance for novices.

The following research questions have guided the development of DCC Sidekick.

- RQ1. How do students choose to engage with a conversational explanation interface when resolving programming errors?
- RQ2. How readable do novice programmers find the conversational explanation interface in comparison to standard compiler messages?
- RQ3. How do students benefit from the conversational explanation interface in resolving errors?

2 DCC SIDEKICK

DCC Sidekick integrates into the open-source Debugging C Compiler (DCC) to utilise its comprehensive run-time and compile-time error detection [16]. In comparison to a standard C compiler, DCC collects extra program state and error information, which is used to inform the generation of explanations. When DCC detects an error, users are able to directly launch the DCC Sidekick dashboard from

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635483

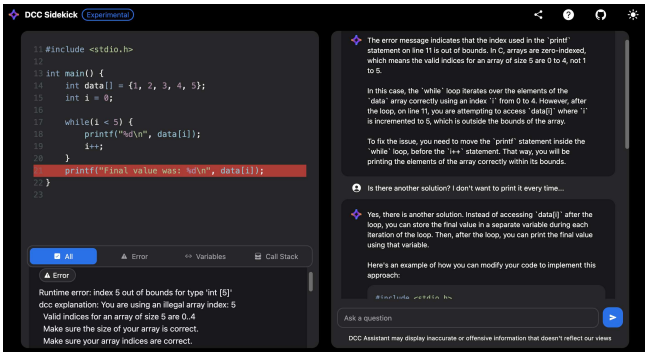


Figure 1: The DCC Sidekick Online Dashboard

the command line. Upon launch, the DCC Sidekick interface, as seen in Figure 1, displays the user's code, standard compiler error message and DCC provided information on one side. This allows users to refer to relevant information without having to divide their attention between development environments. Displayed on the other side is the chat interface which allows students to engage with conversational explanations, generated by querying OpenAI's Chat Completion API (model: *gpt-3.5-turbo*).

When DCC Sidekick is first launched, an initial explanation is generated based on the PEM, user's code, and DCC provided error information, which aims to explain the PEM and guide the student towards understanding the source of the error. The conversational interface can then be used to ask further questions and scaffold understanding of the error. With the aim of ensuring that responses are suitable for unsupervised novice use, DCC Sidekick inserts qualifiers into these questions to discourage the generation of explanations which would provide *copy-paste* fixes for errors.

The DCC Sidekick dashboard only displays information for the first error in the user's code, to reduce the potential confusion from spurious cascading errors, in line with previous work [5].

3 CONTRIBUTIONS

We will be trialling the use of an LLM-based compiler integration in CS1 courses. The purpose of the trial is to investigate whether the conversational explanation interface format with an initial context-rich prompt, all contained within the same development environment has positive impacts for student understanding. During the trial, anonymised usage logs and and survey responses on the readability and helpfulness of the conversational explanation interface will be collected. In addition, we will collect eye tracking, questionnaire and observational data through a series of voluntary usability testing sessions, to investigate the impact of DCC-sidekick in regards to the accessibility, suitability and readability of LLM generated messages for novice programming students as they complete debugging exercises.

We also contribute DCC Sidekick itself, as an open source debugging tool [1], so that it may be used to assist novices in learning to understand PEMs, and to scaffold their understanding of debugging. Ultimately, by investigating the efficacy of DCC Sidekick and the conversational explanation interface it provides, we hope to contribute to the ongoing work in the field of PEM enhancement.

[1]DCC-sidekick implementation: https://github.com/jakerenzella/dcc-llm-explainer

REFERENCES

[1] Titus Barik, Justin Smith, Kevin Lubick, Elisabeth Holmes, Jing Feng, Emerson Murphy-Hill, and Chris Parnin. 2017. Do Developers Read Compiler Error Messages?. In *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017*. Institute of Electrical and Electronics Engineers Inc., 575–585. https://doi.org/10.1109/ICSE.2017.59

[2] Brett A. Becker. 2016. An effective approach to enhancing compiler error messages. In *SIGCSE 2016 - Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. https://doi.org/10.1145/2839509.2844584

[3] Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J. Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter Michael Osera, Janice L. Pearce, and James Prather. 2019. Compiler error messages considered unhelpful: The landscape of text-based programming error message research. In *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*. Association for Computing Machinery, 177–210. https://doi.org/10.1145/3344429.3372508

[4] Brett A. Becker, Kyle Goslin, and Graham Glanville. 2018. The effects of enhanced compiler error messages on a syntax error debugging test. In *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, Vol. 2018-January. Association for Computing Machinery, Inc, 640–645. https://doi.org/10.1145/3159450.3159461

[5] Brett A. Becker, Cormac Murray, Tianyi Tao, Changheng Song, Robert McCartney, and Kate Sanders. 2018. Fix the first, ignore the rest: Dealing with multiple compiler error messages. In *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, Vol. 2018-January. Association for Computing Machinery, Inc, 634–639. https://doi.org/10.1145/3159450.3159453

[6] Michelle Craig and Andrew Petersen. 2016. Student difficulties with pointer concepts in C. *ACM International Conference Proceeding Series* 01-05-February-2016 (2 2016). https://doi.org/10.1145/2843043.2843348

[7] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. 2011. Understanding the syntax barrier for novices. *ITiCSE'11 - Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science* (2011), 208–212. https://doi.org/10.1145/1999747.1999807

[8] Paul Denny, James Prather, and Brett A. Becker. 2020. Error Message Readability and Novice Debugging Performance. In *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*. Association for Computing Machinery, 480–486. https://doi.org/10.1145/3341525.3387384

[9] Paul Denny, James Prather, and Brett A. Becker. 2021. On designing programming error messages for novices: Readability and its constituent factors. In *Conference on Human Factors in Computing Systems - Proceedings*. Association for Computing Machinery. https://doi.org/10.1145/3411764.3445696

[10] L.F. Johnson. 1994. C in the First Course Considered Harmful. *Commun. ACM* 38, 5 (1994), 99–101. https://doi.org/10.1145/203356.203373

[11] Ioannis Karvelas, Annie Li, and Brett A. Becker. 2020. The effects of compilation mechanisms and error message presentation on novice programmer behavior. In *SIGCSE 2020 - Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. https://doi.org/10.1145/3328778.3366882

[12] Tobias Kohn. 2019. The error behind the message: Finding the cause of error messages in python. In *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery, Inc, 524–530. https://doi.org/10.1145/3287324.3287381

[13] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A. Becker. 2023. Using Large Language Models to Enhance Programming Error Messages. In *SIGCSE 2023 - Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, Vol. 1. Association for Computing Machinery, Inc, 563–569. https://doi.org/10.1145/3545945.3569770

[14] Raymond Pettit, John Homer, and Roger Gee. 2017. Do enhanced compiler error messages help students? Results inconclusive. In *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE*. Association for Computing Machinery, 465–470. https://doi.org/10.1145/3017680.3017768

[15] Robert M. Siegfried, Katherine G. Herbert-Berger, Kees Leune, and Jason P. Siegfried. 2021. Trends of commonly used programming languages in CS1 and CS2 learning. In *ICCSE 2021 - IEEE 16th International Conference on Computer Science and Education*. https://doi.org/10.1109/ICCSE51940.2021.9569444

[16] Andrew Taylor, Jake Renzella, and Alexandra Vassar. 2023. Foundations First: Improving C's Viability in Introductory Programming Courses with the Debugging C Compiler. In *SIGCSE 2023 - Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, Vol. 1. https://doi.org/10.1145/3545945.3569768

[17] Jacqueline Whalley, Amber Settle, and Andrew Luxton-Reilly. 2021. Analysis of a Process for Introductory Debugging. In *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3441636.3442300

Why Learn This? Visualizing Pathways between CS Course Topics and Careers

Stacey Levine
Computer Science
Georgia State University
Atlanta, GA USA
slevine4@student.gsu.edu

Anu G. Bourgeois
Computer Science
Georgia State University
Atlanta, GA USA
abourgeois@gsu.edu

ABSTRACT

Computer Science (CS) classes teach technical skills, topics and knowledge areas – often without context where they will be used in future classes. Often times, students routinely struggle and ask questions like "*Why am I learning this*?" and "*What value does it hold?*". In addition to not seeing the correlation of topics, students fail to see what applications these concepts could lead to in the future. When asked what they see themselves doing in the future, we have observed that a significant majority of CS majors respond, "*software developer*".

In this paper, we discuss an ongoing project called Pathways that provides information about their program's CS courses. Upon selecting a course, it will display prerequisites and paths to courses that open with this course, as well as topics covered, skills learned and potential careers tied to these skills. The overall goal is to provide motivation for learning and career planning, with the added benefit of retention, progress, and graduation.

1 INTRODUCTION

Computer Science (CS) curriculum requirements as prescribed by Accreditation Board for Engineering and Technology (ABET) includes technical skills, theoretical concepts, programming, abstraction, and integration and application of computing knowledge. Students often struggle to see relevance in topics and can get discouraged when the future that they see focuses on areas that they lose interest in – typically software development [1, 2]. First-generation students and those without role models that have knowledge or experience in the computing industry are especially at risk of changing majors [3, 4]. The resulting loss of these students continues to exacerbate the issues in lack of diversity. To provide context, Georgia State University (GSU) is an R1 Doctoral University and is designated as a minority-serving institution. Greater than 25% of our undergraduate students are first-generation and enrolls one of the most diverse student bodies in the nation. Making this an extremely relevant and challenging problem for our program.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635605

ACM Reference Format:
Stacey Levine and Anu G. Bourgeois. 2024. Why Learn This? Visualizing Pathways between CS Course Topics and Careers. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY USA. https://doi.org/10.1145/3626253.3635605

institution. Greater than 25% of our undergraduate students are first-generation and enrolls one of the most diverse student bodies in the nation. Making this an extremely relevant and challenging problem for our program.

2 METHODS

In order to help address these issues, we created an application called Pathways. This is a web application that allows the student to visually see the interconnections between classes, while simultaneously being provided with learning outcomes and possible career options. In this way, students will see the relevance of specific topics they are learning in a class and how it is tied to future topics in subsequent courses and ultimately to various career choices. We posit that if students are able to clearly see these connections, they will be better motivated to learn the concepts well.

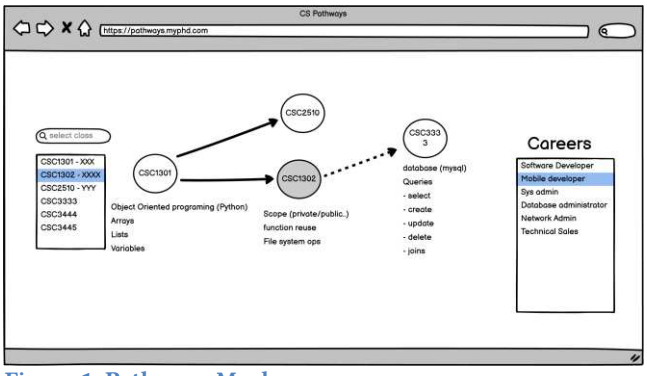


Figure 1: Pathways Mockup

Students also have the option to use the application as a top-down approach. With this option, a user would select a career as their starting point, rather than a course. They would then see the courses and topics that lead to the skills required to help them be best positioned in pursuit of that selected career area.

There is ample research that discusses how the job market is rapidly changing and the skills required, especially in the

computing workforce [5, 6, 7]. Maghsoudi [7] identified 1,315 unique skills that were listed in computing job ads. This is a difficult task for students to navigate what roles they are best suited for, or to determine what skills are needed for which jobs. By providing a bottom-up and top-down approach, students will be able to search based upon their interests, strengths, and experience to determine which classes to take, and/or which careers to pursue.

Supplemental pages in the application allows for:

- Discrete views into each class and the ability to see the topics/outcomes from each class.
- Career listings and the skills/topics that are associated with those careers.

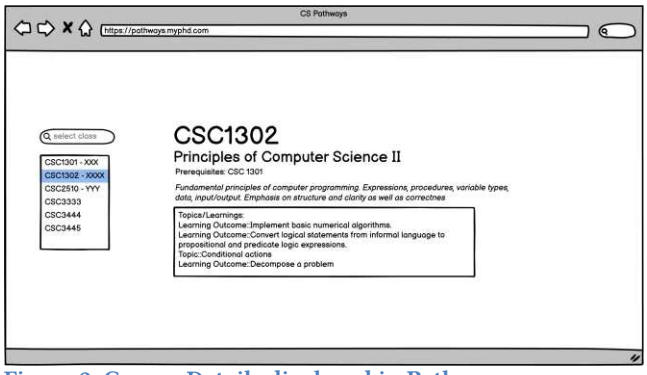


Figure 2: Course Details displayed in Pathways

Course data along with pre-requisites for the courses were manually distilled and entered into a database. Learning outcomes were mapped using ACM guidelines that we utilized from CS Materials [8]. For this initial phase, the mapping of classes to outcomes and careers was done for the initial 3 fundamental courses, specifically:

- CSC 1301 Principles of Computer Science I
- CSC 1302 Principles of Computer Science II
- CSC 2720 Data Structures

The entire catalog for all computer science classes was added for pre-requisite mapping.

To evaluate the impact of Pathways, we will conduct pre- and post-surveys of students in these early classes. The survey will assess their current awareness of career choices and skills that are developed in the courses. Upon completion of the survey, the students will have the opportunity to utilize and explore connections with the Pathways system. Finally, they are asked to take the same survey a second time to determine if there is: an increased awareness in career options; an increased understanding of the relevance of topics being covered in their classes; and an increased motivation to learn and comprehend their course material.

3 FUTURE WORK

Data is currently loaded into the Pathways application through manual efforts and processing of curriculum requirements and syllabi. We are investigating natural language programming techniques and use of large language models, such as SciKit, to automate the parsing of syllabi and course requirement information [9]. This will ease loading of information and maintenance to keep information updated.

For student engagement and longitudinal evaluation of Pathways, we plan to follow the students and evaluate their success rate in retention, progression, and graduation, as well as job placement. We will compare the percentage of students that completed the CS degree and used this system vs those students that did not use the system.

4 ADDITIONAL APPLICATIONS

The goal of this effort was to help increase awareness and retention for first generation computer science students. The application was built in a manner that is discipline agnostic. Assuming that the results continue to be positive, this application could be made freely available for other disciplines and/or institutions to utilize this approach. During some initial user testing, we showed Pathways to a handful of students. Each student had a positive response, with some mentioning that Pathways would prove helpful in planning out their college education.

REFERENCES

[1] Qian, Yizhou, and James Lehman. "Students' misconceptions and other difficulties in introductory programming: A literature review." *ACM Transactions on Computing Education (TOCE)* 18, no. 1 (2017): 1-24.

[2] Peteranetz, Markeya S., Abraham E. Flanigan, Duane F. Shell, and Leen-Kiat Soh. "Career aspirations, perceived instrumentality, and achievement in undergraduate computer science courses." *Contemporary Educational Psychology* 53 (2018): 27-44.

[3] DerSarkissian, Arek, Patricia Cabral, Edward Kim, and Margarita Azmitia. "The high, low, and turning points of college: First generation students' identity negotiations and configurations." *Identity* 22, no. 4 (2022): 265-281.

[4] Toyokawa, Teru, and Chelsie DeWald. "Perceived career barriers and career decidedness of first-generation college students." *The Career Development Quarterly* 68, no. 4 (2020): 332-347.

[5] Deming, David J., and Kadeem Noray. "Earnings dynamics, changing job skills, and STEM careers." *The Quarterly Journal of Economics* 135, no. 4 (2020): 1965-2005.

[6] García-Pérez, Laura, Marina García-Garnica, and Eva María Olmedo-Moreno. "Skills for a working future: How to bring about professional success from the educational setting." Education sciences 11, no. 1 (2021): 27.

[7] Maghsoudi, Mehrdad. "Uncovering the skillsets required in computer science jobs using social network analysis." *Education and Information Technologies* (2023): 1-22.

[8] (n.d.). *CS Materials.* CS Materials. https://cs-materials.herokuapp.com/

[9] B. Ujkani, D. Minkovska and L. Stoyanova, "Using Natural Language Processing for Quality Assurance Purposes in Higher Education," 2021 IV International Conference on High Technology for Sustainable Development (HiTech), Sofia, Bulgaria, 2021, pp. 01-04, doi: 10.1109/HiTech53072.2021.9614206.

Supporting Student Engagement in K-12 AI Education with a Card Game Construction Toolkit

Hansol Lim
hlim5@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Wookhee Min
wmin@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Jessica Vandenberg
jvanden2@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Veronica Cateté
vmcatete@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Judith Uchidiuno
jiou3@gatech.edu
Georgia Institute of Technology
Atlanta, GA, USA

Bradford Mott
bwmott@ncsu.edu
North Carolina State University
Raleigh, NC, USA

ABSTRACT

With the growing prevalence of AI, the need for K-12 AI education is becoming more crucial, which is prompting active research in developing engaging AI learning activities. In this paper, we present our work on a game construction toolkit for middle school students and educators that enables them to tailor an AI-focused unplugged card game activity. In our prior work, we designed, developed, and piloted an unplugged card game activity where players predict the identity of a person based on hand-drawn features extracted from a set of facial cards. The activity aims to teach AI concepts aligned with one of the big ideas in AI utilizing techniques from facial recognition. During our pilot testing of the activity, we discovered that creating face cards that capture students' interest is a crucial factor in promoting student engagement. As a result, we designed a card game construction toolkit that allows students and educators to craft their own face card decks using photos that are personally interesting to them, looking to foster engagement and improve replayability of the activity. The toolkit's design is focused on ensuring easy accessibility and features a simple web-based interface that allows users to download and print their customized cards. We expect this toolkit will enhance the usability and educational effectiveness of our unplugged K-12 AI education activity.

ACM Reference Format:
Hansol Lim, Wookhee Min, Jessica Vandenberg, Veronica Cateté, Judith Uchidiuno, and Bradford Mott. 2024. Supporting Student Engagement in K-12 AI Education with a Card Game Construction Toolkit. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635550

1 INTRODUCTION

Preparing today's K-12 learners for an AI-centric future is both essential and immediate [6]. As the need for K-12 AI education becomes more crucial, initiatives have been undertaken to design, develop, and evaluate unplugged activities for teaching AI concepts [1, 5]. Unplugged activities can support the development of student interest and engagement, while teaching a wide range of AI concepts [4]. For example, Ma et al. [5] introduced two unplugged activities, "The Pasta Land-Decision Tree" and "The *k*-NN-Penguin," which incorporate decision tree and *k*-nearest neighbors algorithms tailored for middle school students. These activities use familiar animals and objects to help students grasp AI concepts in a relatable and age-appropriate way.

In our prior work, we designed, developed, and piloted an unplugged AI learning activity, "Guess Whose Face" [3]. We implemented the activity with 10 rural middle school students during an AI-focused summer camp. Results from exit tickets taken at the end of the activity indicated that the students scored an average of 4.625 out of 5 for enjoyment and engagement and 4.375 out of 5 for conceptual knowledge about *Representation and Reasoning*. Furthermore, we noted that letting students and educators participate in creating and designing part of the unplugged activity could offer promise for further promoting students' engagement. Previous work has explored a variety of toolkits that are available as a means of providing customization opportunities for students [2]. Drawing on the significant potential of unplugged activities for K-12 AI education and informed by our prior findings, we present our work on a game construction toolkit, *Card Craft Studio*, which enables students and educators to customize our "Guess Whose Face" unplugged AI learning activity.

2 MOTIVATION

Prior to designing our toolkit, we developed an unplugged activity called "Guess Whose Face" aligned with one of AI4K12's Big Ideas in AI, *Representation and Reasoning*. Unlike the popular board game "Guess Who?" wherein players ask questions to guess the identity of a person, our game requires students to use features represented as dots extracted from a facial image to guess the person.

The game is designed to be played as a team of two or more students to compete against other teams. There is one *Drawer* and one or more *Guessers* on each team. The role of the *Drawer* is extracting facial features onto tracing paper from a randomly chosen face card without revealing the target face (Figure 1). *Guessers* use a spinner to determine the team's next action and have to recognize the target face from the facial features drawn on the tracing paper. The face card deck serves as a dataset of target faces that the *Guessers* attempt to identify. For the summer camp, we created

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635550



Figure 1: Students playing "Guess Whose Face." Student on the left is the *Guesser* and student on the right is the *Drawer*.

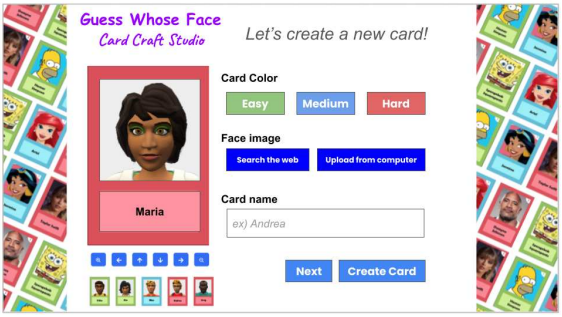


Figure 2: Design of the 'Card Craft Studio' toolkit.

a face card deck with cartoon characters, Disney princesses, and celebrities that represented different difficulty levels.

Through our pilot study with summer camp attendees, we noted diverse student behaviors and collected insightful suggestions that can aid in refining the activity. We observed that students' participation in the activity appeared more enthusiastic when they chose a card of a person or character they knew or were familiar with. Conversely, students seemed to be less enthusiastic when choosing an unknown or unfamiliar card. We collected information about celebrities and characters middle school students prefer through a post-camp survey. The survey suggested that there was a limitation in the original design of the unplugged activity that we could not meet the preferences of all students due to the fixed deck of face cards we provided. One possible solution is to provide a printable DIY (do it yourself) template. This is a well-known method among people who play "Guess Who?" with customized card decks. Using a template supports placing an image file in a fixed number of empty squares that players can print out after mapping all of their own images. Although this method is fairly straightforward, it has the disadvantage of requiring labor-intensive effort to place the images manually. To address this problem, we designed a web-based toolkit allowing students and educators to choose the desired images and automatically arrange them to print their unique face card deck.

3 TOOLKIT DESIGN

The toolkit is designed with simplicity and intuitiveness in mind, aiming to allow first-time users to use it effortlessly (Figure 2). The design of the toolkit emphasized four core functionalities.

Card Configuration. Users can designate a color for the card based on the intended game difficulty levels (Easy, Medium, or Hard). They can also freely select any number of cards to create, ranging from 1 to 20, for customized gameplay.

Card Image Upload. This step facilitates the uploading of facial images for creating cards. Users can either 'drag and drop' an image file or choose one from an existing set of images. By default, the title of the uploaded file will be displayed as the name of the individual shown on the card; however, this can be edited.

Card Production. This feature automatically lays out the uploaded images into a card template, generating a personalized face card deck, and users have the option to print their crafted cards.

Card Sharing and Editing. Users can save their facial images as a project file so that they can later reload it to modify, share with peers, and print it out for future usage.

4 CONCLUSION

Fostering student engagement in K-12 AI educational activities is critical. We have presented our work on a toolkit to enable students and educators to create customized versions of an AI-focused unplugged card game activity to help promote student interest in the activity. The toolkit is being developed as an open-source web-based application that will be available to educators and students interested in using our unplugged activity for K-12 AI education. In the future, it will be important to gather feedback on the toolkit usage from students and educators for future refinements. We aim to collaborate with local communities passionate about AI education, guiding them on how to use the toolkit and card game as we evaluate its effectiveness in practice.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation (NSF) under Grant DRL-2148680. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] Safinah Ali, Vishesh Kumar, and Cynthia Breazeal. 2023. AI Audit: A Card Game to Reflect on Everyday AI Systems. *arXiv preprint arXiv:2305.17910* (2023).

[2] Pedro Beça, Mónica Aresta, Cláudia Ortet, Rita Santos, Ana Isabel Veloso, and Sofia Ribeiro. 2020. Promoting student engagement in the design of digital games: The creation of games using a Toolkit to Game Design. In *2020 IEEE 20th international conference on Advanced Learning Technologies (ICALT)*. IEEE, 98–102.

[3] Hansol Lim, Wookhee Min, Jessica Vandenberg, Veronica Cateté, and Bradford Mott. 2024. Unplugged K-12 AI Learning: Exploring Representation and Reasoning with a Facial Recognition Game. In *Proceedings of the Symposium on Educational Advances in Artificial Intelligence*. AAAI.

[4] Annabel Lindner, Stefan Seegerer, and Ralf Romeike. 2019. Unplugged Activities in the Context of AI. In *Informatics in Schools. New Ideas in School Informatics*, Sergei N. Pozdniakov and Valentina Dagienė (Eds.). Springer International Publishing, 123–135.

[5] Ruizhe Ma, Ismaila Temitayo Sanusi, Vaishali Mahipal, Joseph E Gonzales, and Fred G Martin. 2023. Developing machine learning algorithm literacy with novel plugged and unplugged approaches. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 298–304.

[6] Ning Wang and James Lester. 2023. K-12 Education in the Age of AI: A Call to Action for K-12 AI Literacy. *International Journal of Artificial Intelligence in Education* 33, 2 (2023), 228–232.

Increasing Access to CS Instruction in Low-Income Afterschool Settings

Jared Lim
jlim419@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Grace Barkhuff
grace.barkhuff@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Judith Uchidiuno
jiou3@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

ABSTRACT

Afterschool programs routinely provide or supplement Computer Science (CS) education to K-12 students in the United States. However, such opportunities are rarely available to students from low-income communities due to limited computing resources and CS instructors. One strategy for increasing access to instructors in such settings is to train afterschool staff to teach CS to students in their programs. To investigate the viability of this approach, we recruited and trained three afterschool staff with little prior computing experience to teach their students block programming. We analyzed data from teaching observations and survey-based teaching reflections to understand high-level challenges to this strategy. We found that instructors internalized students' successes and failures as reflections of their own CS teaching ability, and they sometimes struggled to adapt the structured teaching materials to the informal nature of afterschool programs. We reflect on the consequences of these insights and suggest strategies to increase access to CS instructors in low-income afterschool settings.

ACM Reference Format:
Jared Lim, Grace Barkhuff, and Judith Uchidiuno. 2024. Increasing Access to CS Instruction in Low-Income Afterschool Settings. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635559

1 INTRODUCTION

To supplement formal school-based instruction, afterschool programs have increasingly offered CS and STEM opportunities to students [7]. Prior literature has found that these programs can be effective in stimulating interest in a CS career in youth from low-income backgrounds and can overcome issues that prevent students from pursuing CS such as stereotype threat [3].

The increased demand for CS education among K-12 students has led to an increased demand for CS teachers. In addition to increased opportunities for CS teacher certifications, companies such as Microsoft run professional development programs to train computing industry professionals to act as volunteer instructors in high school classrooms [2]. While such programs are successful in addressing the demand for teachers [2], a challenge to this approach is that such training programs are based on adaptations to Advanced Placement (AP) CS curricula, which have an over-representation in White and Asian male students [4]. Opportunities to learn AP CS in formal classrooms can be challenging for low-income communities due to limited computing resources [6]. Programs that focus on utilizing computing professionals and experts to teach computing to youth may not adequately address the unique needs of low-income communities in their training programs and deployment.

To explore strategies to increase CS instructors in low-income settings for younger students, we trained afterschool staff members who worked as homework helpers and program coordinators, with little to no computing knowledge, to teach CS to youth in their programs. This paper provides insights into their experience learning and teaching CS to youth and provides suggestions for improving CS education in low-income afterschool settings.

2 METHODS

We recruited three afterschool staff members from low-income neighborhoods with minimal prior computing experience to teach block programming using code.org's introductory block programming curriculum to their communities.

These instructors completed 10 hours of training, including code.org's CS Fundamentals Express course, and co-worked with researchers in our group to address any areas of ambiguity. They were scheduled to teach twice a week, each session lasting 60 minutes, but their actual teaching frequency depended on their individual schedules and student availability. After every session, they submitted teaching reflections where they shared their favorite and most challenging moments of each session, changes to their teaching approach, and student engagement with the materials. Additionally, we observed each instructor on their first day of teaching to gain insights into their teaching style and the resources available in their afterschool programs.

We conducted inductive thematic analyses on their teaching reflection responses with all themes agreed upon by the team [5]. These themes were triangulated with other data sources such as intake surveys and pictures taken from the afterschool programs. Our research was approved by our university's Institutional Review Board (IRB).

3 RESULTS

1. Volunteer instructors internalize students' successes and failures as a reflection of their own CS teaching ability. We found that the instructors adapted their teaching style based on how students engaged with the material. When a negative learning moment occurred (i.e., students were not engaging with the material or were distracted), instructors interpreted it as reflecting their CS teaching ability.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635559

Table 1: Instructors' self-reported demographics, student demographics, class sizes, and number of sessions taught

Instructor	CS Experience	Teaching Experience	CS Teaching Experience	Student Demographics	# Students / Session	Sessions Taught
Instructor A	None	Yes	None	Black girls, aged 11-13	25-37	5
Instructor B	Basic Python	Yes	None	Black students aged 9-10, majority boys	1-14	12
Instructor C	None	No; longtime afterschool tutor	None	Students, grades 3-8	6-10	9

> Instructor B: *"A challenge was getting my students to not have side conversations while I was teaching and logging in a time-efficient manner... [In the future] I plan on being more stern but still respectful of my students when they are talking while I am teaching."*

In contrast, when a positive learning moment (i.e., students successfully engaging with the material) occurred, instructors felt more confident that they didn't need to change their teaching style.

> Instructor C: *"[My favorite moment in teaching was] seeing the excitement in the students' faces when they figured something out... [Would you change any approach to teaching?] Not at the moment."*

These insights provide evidence that instructors internalized students' successes and failures with the material, feeling that they needed to adapt their teaching style as a result, despite many other factors that might explain students' behaviors.

2. Teaching structured content in an informal afterschool setting is challenging.

Unlike formal school settings, students are not required to attend afterschool enrichment programs. Instructors were able to teach up to 12 sessions, but the program's optional nature meant that they often did not teach the program if they had competing responsibilities on their schedule. This led to a large variance in student and instructor attendance. Even for students who attended sessions, there were other demands for their attention such as recreational opportunities and other play-based programs.

For example, Instructor B says:

> Instructor B: *"I had a two new students join my class, so I had to backtrack on some of the information that I already had given to the students ahead."*

This variance in attendance made it difficult for instructors to build students' knowledge from prior sessions or cover enough material to allow students to engage with CS activities independently.

4 DISCUSSION AND CONCLUSIONS

Our results suggest that instructors internalize students' successes and failures in learning as a reflection of their own CS teaching ability, despite other factors that may explain students' disengagement. This theme from the reflections aligns with Anderson and Page, who in 1995, observed four distinct areas of knowledge in teachers: *technical knowledge*, *local knowledge* (knowledge of local politics and cultures), *craft knowledge*, and *personal knowledge* [1]. The volunteer instructors all have very high local knowledge since they are familiar with their communities but low technical knowledge because they have little to no prior computing background. As a result, they may see students' lack of achievement as a reflection of themselves, given their lack of mastery of the domain content. These instructors were likely over-correcting their teaching style from students' feedback while paying less attention to the content and improving their mastery of CS topics. This result implies that in addition to educating volunteers on CS topics, they should receive training on how to navigate teaching in afterschool settings, properly interpret students' feedback, and individually improve their mastery of the content outside formal training settings.

In addition, teaching structured content in informal afterschool settings comes with several challenges that affect how much students learn. A more viable approach might be to structure instruction around content contained within the allotted time, e.g., 1 hour, without requiring subsequent sessions to depend on instructors' prior teaching. This might lead to a slower rate of learning for students since it's hard to build upon prior knowledge, but continuous exposure to CS instruction in different modalities is likely to have an overall positive effect. Informal afterschool CS education has the potential to reach many underserviced student communities, and we recommend the incorporation of more play-based CS instruction to increase the likelihood of students avoiding CS enrichment programs in favor of other play-based enrichment opportunities.

REFERENCES

[1] Robert Donmoyer, Michael Imber, James Joseph Scheurich, Gary L Anderson, and Bonnie Page. 1995. *Narrative Knowledge and Educational Administration: The Stories that Guide Our Practice.* State University of New York Press, 124–134.

[2] Nathaniel Granor, Leigh Ann DeLyser, and Kevin Wang. 2016. TEALS: Teacher Professional Development Using Industry Volunteers. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 60–65. https://doi.org/10.1145/2839509.2844589

[3] Linda S. Hirsch, Suzanne Berliner-Heyman, Rosa Cano, Howard Kimmel, and John Carpinelli. 2011. Middle school girls' perceptions of engineers before and after a female only summer enrichment program. In *2011 Frontiers in Education Conference (FIE)*. S2D–1–S2D–6. https://doi.org/10.1109/FIE.2011.6142990

[4] Kip Lim and Colleen M. Lewis. 2020. Three Metrics of Success for High School CSforAll Initiatives: Demographic Patterns from 2003 to 2019 on Advanced Placement Computer Science Exams. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 598–604. https://doi.org/10.1145/3328778.3366810

[5] Anselm Strauss and Juliet Corbin. 1998. Basics of qualitative research techniques. (1998).

[6] Ryan Torbey. 2023. Inequities of Enrollment: A Quantitative Analysis of Participation in High School Computer Science Coursework Across a 4-Year Period. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1* (Chicago, IL, USA) *(ICER '23)*. Association for Computing Machinery, New York, NY, USA, 344–355. https://doi.org/10.1145/3568813.3600125

[7] Judith Odili Uchidiuno, Jaemarie Solyst, Erik Harpstead, and Ross Higashi. 2023. "I'm a Little Less Inclined to Do It": How Afterschool Programs' Culture Impact Co-Design Processes and Outcomes. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) *(DIS '23)*. Association for Computing Machinery, New York, NY, USA, 1263–1276. https://doi.org/10.1145/3563657.3596006

Teaching Algorithm Design: A Literature Review

Jonathan Liu
jonliu@uchicago.edu
University of Chicago
Chicago, Illinois, USA

Seth Poulsen
seth.poulsen@usu.edu
Utah State University
Logan, Utah, USA

Hongxuan Chen
hc10@illinois.edu
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA

Grace Williams
gcwill@uchicago.edu
University of Chicago
Chicago, Illinois, USA

Yael Gertner
ygertner@illinois.edu
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA

Diana Franklin
dmfranklin@uchicago.edu
University of Chicago
Chicago, Illinois, USA

ABSTRACT

Algorithm design is a vital skill developed in most undergraduate Computer Science (CS) programs, but few research studies focus on pedagogy related to Algorithms coursework. To understand the work that has been done in the area, we present a systematic survey and characterization of existing studies in the CS Education literature related to the teaching of algorithm design at the undergraduate level. Across all papers in the ACM Digital Library, we only find 97 applicable papers. We classify these papers by topic, evaluation metric, evaluation methods, and intervention target. We present the results of these classifications alongside insights about existing knowledge, rigor, and contribution rates. We hope that this work not only provides a detailed representation of the current corpus of CS Education work related to algorithm design but also demonstrates that the body of knowledge is sparse and supports further research in the area. For future work, we intend to investigate and synthesize the conclusions reached by these papers.

ACM Reference Format:
Jonathan Liu, Seth Poulsen, Hongxuan Chen, Grace Williams, Yael Gertner, and Diana Franklin. 2024. Teaching Algorithm Design: A Literature Review. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635548

1 INTRODUCTION

The study of algorithms is widely regarded as a critical part of an undergraduate Computer Science (CS) degree [2] and has clear practical applications, especially in software-related careers [1]. Though the design of rudimentary algorithms has been extensively researched in an introductory programming context, most students will take a more in-depth algorithms-focused course later in their degree. Research on algorithm design instruction, as taught in upper-level algorithms courses, is critically underrepresented in the literature. As such, many open questions remain about student experiences and difficulties in learning this material, as well as pedagogical practices for teaching algorithm design in an effective and equitable manner.

In this poster, we present our first results from a systematic survey of CS Education literature related to algorithm design education, following the guidelines from [3]. We provide a characterization of the literature, answering the following research questions:

- **RQ1** What algorithm topics are currently represented in the literature, and what kinds of studies are being conducted on these topics?
- **RQ2** What kinds of interventions have been developed to teach algorithm design?

2 METHODS

2.1 Search Strategy

We conducted our search on the ACM Full-Text Collection in the ACM Digital Library (DL). To capture our research questions with an appropriate query, we used the most recent ACM Curricular Guidelines [2] and results from a recent survey of Algorithms courses [4] to identify the set of topics that could be considered appropriate for an Algorithms course. For each topic, we designed a query to capture the corpus of literature on the topic. With this set of queries, we searched the ACM DL for all articles from venues sponsored by SIGCSE between 1970 and July 2023 (820 unique entries) and then used the same queries for ToCE (56 entries).

2.2 Study Selection

From this set, we omitted papers not relevant to algorithms topics and papers that do not present data from an evaluation on university-level students. Authors reached high inter-rater agreement on these criteria before beginning to tag independently.

2.3 Paper Tagging

We then developed a set of tags for classifying the relevant papers. Each paper was tagged by the topic(s) it focused on, and an extra *Not topic-specific* tag was added for papers that considered Algorithms courses as a whole. Tags were also developed to capture the research methods and metrics by which data was evaluated. Interventions were then classified by the aspect of the course affected. The tagging list does not exhaustively consider every possibility for evaluation and intervention methods, but it captured every paper found in our review. Authors also reached high inter-rater agreement with these tags, and tagged most of the papers independently, discussing with each other when unsure.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635548

3 RESULTS OVERVIEW

The full dataset can be found at https://doi.org/10.7910/DVN/KTR2ZH. We present a high-level overview of our findings here. Table 1 provides information on the number of papers within each topic that were tagged in each category. Note that a paper with multiple topics would be represented in multiple rows and columns. The last row contains a summary of the results.

In the over 50-year span from 1970-2023 covered by our queries, we only found 48 papers that study, with data, pedagogy related to *any* specific topic in algorithm design. Even the most-covered topics have no more than a dozen papers, and some important topics like Divide-and-Conquer have seen incredibly little research despite being taught in more than 80% of Algorithms courses [4].

Nearly every paper meeting our selection criteria presents quantitative data, and almost 75% of papers use student scores and grades. Around 40% of the papers run statistical tests, but less than 25% run controlled trials. On the other hand, almost half of the papers found use qualitative data. Thematic coding, found in only 16 papers, was done for interview responses, student work, and survey responses.

A quarter of the papers surveyed do not develop an intervention. Many of these papers analyze student problem-solving strategies when presented with algorithm design problems, and a number of others study topic-specific misconceptions. Of the interventions, more than 40% center around a tool, and over a third involve changes to student independent work.

We also find that SIGCSE's largest conferences, the Technical Symposium and ITiCSE, account for a vast majority of the papers (44 and 33, respectively) in our literature review, with ICER contributing nine. We found five relevant papers from ToCE (and its previous moniker JERIC) as well.

Among the authors represented by the papers in our review, we find only six authors who have published 3 or more papers, 11 who have published 2, and 191 authors who have only published one paper in our review.

4 FUTURE WORK

This poster provides results from our classification of papers in Algorithms Design by topic and methodology. For future work, we hope to focus on synthesizing results from papers. This can be done in topics with a sufficient number of papers, and with the body of papers tagged as "Not topic-specific," which represents a broad set of ideas that will require further study to glean insight from.

More broadly, future research in the community can use our findings to focus their research on under-studied topics and approaches. For example, there are notably few papers on topics like Shortest-path Algorithms, despite their broad prevalence, and few papers studying exam-focused interventions.

REFERENCES

[1] Mahnaz Behroozi, Chris Parnin, and Titus Barik. 2019. Hiring is broken: What do developers say about technical interviews?. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–9.

[2] Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* ACM.

[3] Barbara Kitchenham and Stuart Charters. 2007. *Guidelines for performing Systematic Literature Reviews in Software Engineering.* Technical Report EBSE-2007-01. Software Engineering Group, Keele University and Department of Computer Science, University of Durham, Salt Lake City, UT.

[4] Michael Luu, Matthew Ferland, Varun Nagaraj Rao, Arushi Arora, Randy Huynh, Frederick Reiber, Jennifer Wong-Ma, and Michael Shindler. 2023. What is an Algorithms Course? Survey Results of Introductory Undergraduate Algorithms Courses in the U.S.. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. ACM, 284–290.

Table 1: Tags by Topic. Note that some papers contribute to multiple rows in the table. Some topics omitted for brevity.

		Evaluation								Metric			Intervention							
Topic	Total	Quantitative Data	Survey	Scores/Grades	Statistical Tests	Controlled Trial	Qualitative Data	Interview	Thematic Coding	Performance	Affect	Persistence	None	Tools	Course Policy	Content Presentation	Discussion/Lab	Student Work	Exams
Depth- and Breadth-First Traversals	7	6	5	4	3	3	3	1	1	4	5	0	0	3	0	2	1	3	0
Divide-and-Conquer	3	3	1	3	2	1	2	1	0	3	0	0	2	1	0	0	1	0	0
Dynamic Programming	10	10	6	8	5	3	5	2	1	8	5	1	3	4	0	1	1	3	0
Greedy Algorithms	8	8	4	6	0	0	6	2	5	6	4	0	2	3	0	3	2	2	0
Minimum Spanning Trees	5	5	4	3	1	1	2	0	0	3	3	1	0	3	0	0	1	4	1
Reductions	6	5	3	6	1	0	4	1	2	6	2	0	3	0	1	2	0	2	0
Representations of Graphs	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shortest-path Algorithms	6	6	4	5	2	2	1	0	0	5	4	1	0	6	0	1	1	1	0
$O(n \log n)$ Sorting Algorithms	12	10	7	7	5	3	7	1	1	7	10	1	2	4	0	2	2	4	0
Total (at least one topic)	48	44	28	36	15	11	25	6	9	36	28	2	10	20	1	10	9	15	1
Not topic-specific	49	46	27	34	23	7	22	6	7	34	31	2	11	16	11	9	7	16	2
Total	97	90	55	70	38	18	47	12	16	70	59	4	21	36	12	19	16	31	3

The Current Research Landscape of Computing Education in Elementary Settings: A Systematic Literature Review

Ruohan Liu
University of Virginia
ces8hz@virginia.edu

ABSTRACT

Despite the proliferation of research studies examining elementary computing education worldwide, knowledge regarding the implementation and outcomes of elementary computing education is limited. In this study, we systematically reviewed 45 empirical papers published between January 2015 and December 2021 to establish a holistic understanding of recent research trends, implementation practices, and student outcomes in elementary computing instruction. This review identified three areas of focus that the current literature has explored: 1) curriculum and lesson development; 2) computing tools and learning environment design; and 3) computing pedagogy. In terms of implementation, we identified 1) eight types of subject contexts where computing instruction is implemented; 2) seven types of technologies that are used to engage elementary students in computing; and 3) eight types of learning activities that are implemented. Regarding student outcomes, the current literature reported student outcomes in six dimensions. This review presents the current research landscape of elementary computing education in terms of research trend, implementation practices, and student accomplishment. It showcases the versatility of instructional practices, highlights the progress of current efforts, while also pinpoints potential issues and concerns for future improvement.

ACM Reference Format:
Ruohan Liu. 2024. The Current Research Landscape of Computing Education in Elementary Settings: A Systematic Literature Review. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635622

1 INTRODUCTION

To ensure children are fully prepared for the future, it is essential that they are offered opportunities to learn computing knowledge and skills at an early age. As information, media, and technology skills become a core component of 21st century skills [5], many countries have launched initiatives and national curricula to support computing education in elementary schools.

Driven by the worldwide efforts to promote computing education at the elementary levels, research studies exploring the implementation and outcomes of computing instruction in elementary grades have proliferated. There are prior reviews investigating elementary computing education in multiple dimensions, such as instruction of computational thinking [2, 4]; student participation and barriers in CS learning [9], and implementation of computing technology[7]. Building on the existing reviews, this study extends to a broader field of computing that encompasses CS, programming, and CT. Furthermore, it aims to present a more recent and comprehensive landscape of elementary computing education in terms of research focus, instructional practices, and student outcomes, which has not been widely examined in prior reviews. Specifically, the following research questions guided this study:

- What research topics related to elementary computing instruction are examined in the current literature?
- How is computing instruction implemented in elementary grades?
- What student outcomes are reported in the literature?

2 METHOD

Following the systematic review procedures [6], this review was conducted through four stages: article identification, article searching and screening, data extraction and monitoring, data synthesis.

Article Identification. We searched for articles in six databases: Education Source, ERIC, Academic Search Premier, APA PsycInfo, ScienceDirect, JSTOR as these are well-established research repositories that have been applied in existing CS education reviews [10]. To expand the data pool, We also searched three CS education conference proceedings: SIGCSE, ItiCSE, ICER.

Searching and Screening We used the searching string: "(computer science or computational thinking or programming) AND (elementary education or primary education)" to identify relevant articles. Screening criteria, aligned with the research questions, are applied (Figure 1). Generally, out of the 1689 articles identified initially, 45 met our screening criteria and were retained for final analysis.

Data extraction, monitoring, and synthesis. Based on the research questions, two coders involved in this study developed an

Inclusion Criteria	Exclusion Criteria
1. Timespan: 2015-2021; 2. Peer-reviewed empirical studies; 3. Students in K-6 grades as the studied population; 4. Studies focusing on elementary computing instruction; 5. Studies with sufficient information about computing instructional practice and learning process; 6. Available in full-text; 7. Written in English.	1. Abstract/work in progress; 2. Conceptual and theoretical papers; 3. Studies targeting non-elementary participants; 4. Studies irrelevant to computing instruction; 5. Studies in clinical settings.

Figure 1: Screening Criteria

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635622

initial coding scheme. This scheme was iteratively discussed and refined during the initial coding phase and was finalized once the coders reached consensus and felt confident about it. Data extraction was performed by the two coders collaboratively, the extracted data were discussed and compared iteratively to ensure that the coders extracted information in a consistent manner. Any uncertainties and disagreements regarding the extracted information were resolved through rigorous discussions. Data analysis was performed using thematic analysis approach [1].

3 RESULTS

3.1 RQ1. What research topics related to elementary computing instruction are examined in the current literature?

We identified three areas of focus that the current literature explored: 1) computing technology and learning environments (n=14); 2) computing pedagogy (n=11); 3) computing curriculum and lesson development (n=20). A main focus of the reviewed studies is about developing and implementing, and evaluating elementary computing curriculum and lessons. The relevant studies adopted established (e.g., national CS curricula, open-source curricula) or self-designed courses to engage participants in computing over a period of time, and examined their learning outcomes in areas such as computing knowledge and skills, self-efficacy, attitudes, and general cognitive abilities (e.g., [8]).

Another main topic is developing tools and computing environments for elementary students. Those relevant studies explored the effects of emerging technology and computing tools on participants' engagement, attitudes, and learning outcomes of computing knowledge and skills (e.g., [3]). A number of studies focused on computing pedagogy, including 1) instructional techniques for teaching CS concepts and knowledge; 2) pedagogical models for instructional activities; 3) interdisciplinary computing pedagogy; and 4) strategies for engaging students and promoting interactions in computing learning.

3.2 RQ2.How is computing instruction implemented in elementary grades?

We identified eight subject areas in which computing instruction is implemented: CS/programming, science, math, language and literacy, STEM, Engineering, Arts, and Social studies.

In terms of learning technology applied, we identified twenty-five tools that feature seven genres: 1) visual block-based programming, 2) text-based programming, 3) tangible programming, 4) games, 5) digital manipulative, 6) programming robotics, and 7) immersive virtual simulation.

In terms of learning activities implemented, we identified eight types of activities to engage students in computing: 1) hands-on exploration, 2) learning with explicit instruction and modeling; 3)structured and open-ended problem-solving; 4) game-based learning; 5) project-based learning; 6) collaborative and cooperative learning; 7) embodied learning; and 8) story creation and storytelling.

3.3 RQ3. What student outcomes are reported in the literature?

The reviewed studies highlighted student outcomes across six dimensions: 1) computing performance and skills; 2) affective outcomes; 3) computing content knowledge; 4) disciplinary learning outcomes; 5) general cognitive skills; and 6) interdisciplinary learning outcomes. Notebly, students' development of computing performance and skills, affective outcomes in computing, and acquisition of computing content knowledge are emphasized more frequently in the literature. A few studies identified that computing experience is beneficial to students' learning in other disciplines, promotes students' general cognitive skills, and enriches their interdisciplinary learning experiences.

4 CONTRIBUTION AND FUTURE WORK

Findings of this review underscore the synergistic impact of accessible computing curricula, supportive environment of learning and technology, and cognitively appropriate pedagogy on promoting elementary students' participation in computing. Given that this review is still in progress, we will provide more in-depth implications for future research and practice in our forthcoming presentation.

REFERENCES

[1] Virginia Braun and Victoria Clarke. 2012. *Thematic analysis.* American Psychological Association.

[2] Janne Fagerlund, Päivi Häkkinen, Mikko Vesisenaho, and Jouni Viiri. 2021. Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education* 29, 1 (2021), 12–28.

[3] Anna Gardeli and Spyros Vosinakis. 2019. ARQuest: A tangible augmented reality approach to developing computational thinking skills. In *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*. IEEE, 1–8.

[4] Panagiotis Kakavas and Francesco C Ugolini. 2019. Computational thinking in primary education: A systematic literature review. *Research on Education and Media* 11, 2 (2019), 64–94.

[5] Ken Kay and Valerie Greenhill. 2010. Twenty-first century students need 21st century skills. In *Bringing schools into the 21st century*. Springer, 41–65.

[6] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology* 51, 1 (2009), 7–15.

[7] Ruohan Liu, Feiya Luo, and Maya Israel. 2023. Technology-Integrated Computing Education in Early Childhood: A Systematic Literature Review. *Journal of Educational Computing Research* (2023), 07356331231170383.

[8] Taras Panskyi, Zdzislawa Rowinska, and Sebastian Biedron. 2019. Out-of-school assistance in the teaching of visual creative programming in the game-based environment–Case study: Poland. *Thinking Skills and Creativity* 34 (2019), 100593.

[9] Umar Shehzad, Mimi Recker, and Jody Clarke-Midura. 2023. A Literature Review Examining Broadening Participation in Upper Elementary CS Education. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 570–576.

[10] Xiaodan Tang, Yue Yin, Qiao Lin, Roxana Hadad, and Xiaoming Zhai. 2020. Assessing computational thinking: A systematic review of empirical studies. *Computers & Education* 148 (2020), 103798.

Circuit Tiles

Mixing Culture, Crafts, and Circuits to Inspire Diversity in Computing

Maria Lopez-Delgado
l113@umbc.edu
Information Systems
University of Maryland, Baltimore County
Baltimore, Maryland, US

ABSTRACT

Circuit Tiles is a constructionist kit that utilizes children's curiosity about mosaics to explore STEAM skills and topics. The kit, targeted to children ages six or older, aims to discover basic circuits and electronics through modular circuit tiles by presenting them in a different context. Utilizing an artistic context, drawing from Spanish-style tiles and mosaic art traditions in Latin American and Hispanic countries allows an underrepresented group in STEAM to experience essential ideas differently. The design aims to increase the diversity of low-floor tangible electronic materials.

ACM Reference Format:
Maria Lopez-Delgado. 2024. Circuit Tiles: Mixing Culture, Crafts, and Circuits to Inspire Diversity in Computing. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635558

1 INTRODUCTION

Imagine if you could build a mosaic you could interact with. But how? By adding LEDs, push buttons, or switches. This paper explores the creation of Circuit Tiles, a constructionist kit that utilizes children's curiosity about mosaic art as a tool for them to explore STEAM skills and topics. The design aims to increase the diversity of low-floor tangible electronic materials targeted to children six or older. The kit's target audience is children within the community who otherwise may not have the opportunity to experience these ideas. The design also aims to create an artifact that captures some of the design principles used in the manipulatives developed by Froebel and Montessori.

Further, Circuit Tiles allow them to see and develop STEAM skills in a different context. They experience the intersection between art and technology. Moreover, it can enable them to learn the skills needed for the 21st century, particularly problem-solving and creative thinking. This "hybrid craft" [4] aims to expose the critical ideas of circuitry to novel learners in a culturally creative context. Throughout the design of the kit, crucial decisions were made regarding background elements, like the user not having to physically create connections with the conductive lines and the electronic elements. Rather, designed to provide a manageable way of making the circuit accessible for novice learners.

Many have contributed to exploring similar ways to present these ideas and skills, ranging from the Lectron System [5], Snap circuits [6], littleBits [3] or LightUp [1]. Taking inspiration from the work done with similar terms like "hybrid crafts" and "creative technologies" in regards to target need, audience, and inspiration. Circuit Tiles add to the discussion by focusing on the inclusion of culture, and creativity to reach other children to learn about electronics.

2 DESIGN

But why use tiles? A crucial decision was to consider the child's culture and prior experiences. The consideration of culture as part of the design of a constructionist tool serves as a foundation during the process [7]. Further, as mentioned by Hooper (1998), as cited by Pinkett (2000), the use of culture while making is an opportunity to help increase the interest of the participants [8]. For this reason, the tradition of mosaic art and Spanish-style tiles culture is an excellent resource for children to create meaningful art and learn about circuits.

Although traditional mosaics use ceramic or crystal tiles, the entire kit is made of wood, a familiar material for users. This design decision is similar to the one made by the KinderLab Robotics Lab when creating the tangle programming wooden blocks for their KIBO robot [2]. Additional reasoning is that it is frequently used in educational toys influenced by Maria Montessori or Friedrich Froebel. Like the kit, their toys promote clear concepts and allow for manipulation that permits successful learning, construction, and design with the primitives and modular that encourage inquiry and trial [9].

The board delimits the physical space the child has to build their mosaic. Its dimensions are 6" x 6", and it is subdivided into a grid of 2" x 2". The grid has nine small squares, permitting the board to have an explicit center. This is helpful for the planner when designing their mosaics and the "bricoleur", which creates their own methods while interacting with the tiles and creating their art pieces. Additionally, smaller holes allow the board to work as a pegboard when used with the mosaic pieces. This mechanism helps the tiles have stability and security. This is important for the incorporation of the circuits. Also, this mechanism is comparable to gluing the tiles when creating a mosaic.

The other elements are the tiles with two shapes and dimensions. There is a square (1" x 1") and a rectangular piece (2" x 1"). Mimics the nature of mosaic making, where artists work tiles of different shapes and dimensions. The tiles are painted in bright primary

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635558

colors. Similar to the superficial aesthetics used by Montessori and Froebel, which are attractive and "aligned with the conceptual design" [9]. Each piece has one or two small pieces underneath that connect to the board pegboard system. The top of the tiles can be empty or have the circuit components. Further, this allows the child to use the kit in the intended ways to create art, build circuits, or join both.

As for the use context, it is suitable for an informal education setting, particularly, an at-home art activity. There is also the opportunity to use it in more formal educational settings and collaborate with educators on how to best implement the kit.



Figure 1: Circuit Tiles Elements

The minimalist approach for the circuit elements aligns with Montessori's and Froebel's principles. Additionally, they are not a prevalent portion of patterns or figures created by the child. For the construction of the circuits, users have pieces with LEDs, circuit lines, push buttons, or switches. The white lights will allow the current flow to be the same and not have an additional bug because of the current resistance. Another design choice was not to identify the positive or negative sides of the LEDs. Enabling users to tinker and debug their circuit. Developing debugging skills helps the user identify issues and possible problem-solving strategies for working with similar concepts in the future.

3 DISCUSSION

The design highlights a few elements the user will focus on and help with their learning experience. For example, given the primitive's design, particularly the ones with the circuit element, the child only has to focus on creating the sequence. This makes building the circuit simpler for beginners and enables them to notice the elements of their design. It permits the user to detect the parallel conductive lines and observe the similar nature of their circuit. Focusing on the connections between the elements helps debug the circuit. Although they may have to debug their circuit, it will allow them to troubleshoot possible reasons why it is not working and strategies to fix it. Exposing the user to learn these skills for future activities is essential.

Some elements of the design were highlighted, and some were disguised to help focus on the intended learning objectives. For example, the board and tiles pegboard design constrain the position of the tiles, increasing the user's chance of success when building their mosaic. Another backgrounded element was that the user would not have to focus on physically creating the connections with the conductive lines and the electronic elements in the initial kit, providing a manageable way of making the circuit accessible for novice learners. Another design decision that led to a background element was not labeling the sides of the LEDs, implying that the child must tinker and debug their circuit. Because of the nature of the tiles' material, they can choose to label it themselves. Learners need to use this tool to learn the basic ideas behind electronics and use them when working with similar concepts.

All these decisions implied a series of trade-offs. For example, not knowing how many tiles the user will need with and without circuit components to complete their design. It is challenging to predict all the possible combinations and create tiles that allow all the design. Similarly, many tiles only contain parallel conductive material without other elements. Initially, the intended learning objective was for the user to create circuits in parallel and series, forcing the design to only allow for circuits in parallel. As for the toy's prototyping, making the tiles with the necessary precision for the circuitry elements to align correctly was time-consuming.

4 CONCLUSION

All in all, Circuit Tiles is a constructionist kit that aims to help the accessibility gap of experience regarding maker experiences and electronic knowledge. The gap also includes the lack of accessibility of types of experiences to children and the lack of representation in the field. Circuit Tiles attempted to address these needs by creating a kit that showcased electronics in a different context, related to culture, and created with low-cost materials, making it accessible to most. Future work includes a play-test study to understand how the use of a constructionist kit could spark girls' interest to learn more about electronics and apply essential skills related to STEAM.

ACKNOWLEDGMENTS

I want to thank Dr. Nathan Holbert, from Teacher College - Columbia University for helping me understand how to design and create constructionist toys and for all the support to bring this idea to life. Also thank you to my advisor Dr. Patricia Ordonez for encouraging me to publish and guiding me in the process.

REFERENCES

[1] Zain Asgar, Joshua Chan, Chang Liu, and Paulo Blikstein. 2011. LightUp: A Low-Cost, Multi-Age Toolkit for Learning and Prototyping Electronics. In *Proceedings of the 10th International Conference on Interaction Design and Children* (Ann Arbor, Michigan) *(IDC '11)*. Association for Computing Machinery, New York, NY, USA, 225–226. https://doi.org/10.1145/1999030.1999067

[2] Marina Umaschi Bers. 2018. Coding, playgrounds and literacy in early childhood education: The development of KIBO robotics and ScratchJr. *2018 IEEE Global Engineering Education Conference (EDUCON)* (2018). https://doi.org/10.1109/educon.2018.8363498

[3] Stacy Brown. 2018. Big Impact with littleBits. *Library Technology Reports* 54, 4 (2018), 28–31. https://journals.ala.org/index.php/ltr/article/view/6673/8963

[4] Leah Buechley and Hannah Perner-Wilson. 2012. Crafting Technology: Reimagining the Processes, Materials, and Cultures of Electronics. *ACM Trans. Comput.-Hum. Interact.* 19, 3, Article 21 (oct 2012), 21 pages. https://doi.org/10.1145/2362364.2362369

[5] Elmer Carlson. 1967. Lectron. *Electronics Illustrated* (1967). http://www.decodesystems.com/lectron.html

[6] Eleneco Electronics. 2020. Snap Circuits. https://www.elenco.com/snap-circuits/

[7] Alan Kay. 2017. Rethinking Ideas. Video. https://www.media.mit.edu/videos/seymour-2017-01-26/

[8] Randal D Pinkett. 2000. Bridging the digital divide: Sociocultural constructionism and an asset-based approach to community technology and community building. In *81st Annual Meeting of the American Educational Research Association (AERA), New Orleans, LA*. 24–28.

[9] Oren Zuckerman. 2010. Designing digital objects for learning: lessons from Froebel and Montessori. *International Journal of Arts and Technology* (2010). https://doi.org/10.1504/ijart.2010.030497

Leveraging Science Fiction to Prepare Students for Tech Battles

Jessica Lucchetta
University of Trento
Department of Information Engineering and Computer Science
Trento, Italy
jessica.lucchetta@unitn.it

Andrea Saiani
Witted Srl
Trento, Italy
andrea@witted.eco

Milena Stoycheva
Junior Achievement Bulgaria
Sofia, Bulgaria
milena.stoycheva@jabulgaria.org

Lorenzo Angeli
University of Trento
Department of Information Engineering and Computer Science
Trento, Italy
lorenzo.angeli@unitn.it

ABSTRACT

In the last decades, teaching methods in computing are shifting towards active learning. Among these, a notable case are debate-based classes which can be used to develop skills such as critical thinking which, are in high demand in the workplace. The choice of debate topic is a particularly controversial theme, since it can influence the students' motivation, experience, and understanding of the course's content. In this poster, we propose an evolution of a debate-based teaching methodology, called Tech Battles, to include the reading of science fiction short stories as a preparatory activity. The intervention takes place in an Innovation and Entrepreneurship course for computing students at the University of Trento, where science fiction can help students develop critical thinking about technology and innovation, but also aid them in understanding the debate's context, and ultimately enhance their learning. We conclude our reflections by exploring possible implications of this change, connecting it to existing traditions that leverage science fiction as a pedagogical tool.

KEYWORDS

debates; innovation and entrepreneurship; science fiction

ACM Reference Format:
Jessica Lucchetta, Andrea Saiani, Milena Stoycheva, and Lorenzo Angeli. 2024. Leveraging Science Fiction to Prepare Students for Tech Battles. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635532

1 CONTEXT

The focus of most computing curricula tends to be centred on technical knowledge [4]. However, in the last years, the computing field and the related work environment have given increasing importance to competencies and attitudes (again, [4]). These skills are difficult to develop through traditional learning methods [5], leading to a rise in the adoption of active learning approaches.

Debates represent a particularly interesting case within active learning. In-class debates are commonly used in humanities and law, as they promote interaction between students and increase engagement. In the computing domain, however, while literature reports some examples as far back as 2002 [1, 10], examples remain rare. Alfor and Surdu [1] claimed that debates could help students develop skills such as synthesising information, self-directed learning, critical thinking, and verbal communication.

To achieve these gains, though, a key issue is the selection of the debate's topic. A good topic should encourage students to think and deepen their knowledge [1], but should also feature clear opposing sides, easing the students in understanding the distinct positions and drawing different perspectives [7]. Debate topics play an important role in engaging the students and, if they are not appropriate, may cause a loss of motivation [11].

In this contribution, we propose to use science fiction short stories as an aid to create debate topics, and as a preparation activity. We aim to provide a scaffold for the students' understanding of the debate topics, stimulate their curiosity, and ensure that they can get a grasp on the content as early as possible in the process.

2 PEDAGOGICAL CONTEXT

Our intervention takes place within a course called "Innovation and Entrepreneurship (I&E) Basics", taught at the University of Trento. The course teaches basics of innovation theory, entrepreneurial practice, and social impacts of computing, while developing skills such as clear presentation and critical thinking. The course involves first-year Master's computing students in a curriculum that includes a 30-ECTS "minor in I&E" while specialising in a technical subdomain such as Autonomous Systems. The course has been attended by groups as small as 20 students and as large as 180 but, since 2020, attendance has stabilised to around 40 students.

The course notably employs Tech Battles (Battles), a teaching method where two teams of 4–5 students debate opposed views

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635532

on a socio-technical theme in the domain of technological innovation. Since their first appearance in 2013, Battles have evolved from historical case studies to discussions of fictional scenarios [3]. Battles have then been further adapted to make the exercise more structured [2], following a frequent request from the students.

Each Battle involves (1) a Battle preparation meeting; (2) the debate itself; and (3) a "Battle Report". The Battle preparation is a 1-hour meeting involving the two debating teams and the teachers, and is used to set each Battle's rules and scenario. Scenarios aim to build plausible alternative worlds, where the class can have a discussion free from their own prejudices [8]. The Battle, done in a 2-hour class slot, sees the teams debating each other, while the class acts as an active audience, asking questions and ultimately voting for the more convincing proposal. Finally, in the report, the teams synthesize their learnings and reconcile their positions.

The current structure, however, still presents a key weakness: students come to the Battle preparation meeting without knowing anything about the topic of the debate. This, we argue, is a missed opportunity in making the preparation meeting more focused. Furthermore, the creation of scenarios is so far reliant on the teachers' creativity, when fiction literature could provide an abundance of potential scenarios ripe for debate.

3 OUR PROPOSAL

In this contribution, we propose to better prepare students for the Battle preparation meeting through an activity where students read science fiction (SF) short stories. Suitable short stories can introduce students to the themes that they will address during their Battle with a modest time investment. By choosing relevant stories, teachers can "prime" the students' imagination, stimulate their curiosity, and enrol them in co-creating the scenario.

SF has a long tradition of exploring socio-ethical dilemmas using scientific and technological metaphors: SF's power for critique also been noted by Haraway [6] and has been used to create academic syllabi such as Speculating Futures[1]. The stories we adopt in the course explicitly feature critiques to innovation, progress, or specific pieces of technology, since they provide the key benefit of being easy to connected to contemporary technological controversies, such as privacy and AI, and their ease in creating grounds to generate scenarios for debates [9].

In the course, we assign student teams a short story three weeks before their Battle. The ensuing Battle preparation meeting starts by discussing the short story, and then proceeds by co-creating the scenario. With this adaptation, scenario building can now see a more active contribution of the students because, through the short story, they can better grasp the scenario's pedagogical intent. As an example, students can give feedback and help the process of generating the debate positions based on the tensions they found in the short story. The meeting then finishes as per our normal model, with teams being assigned to one of the two positions.

4 CONCLUSIONS/FUTURE WORK

In this contribution, we presented an evolution of Tech Battles which adds a second preparatory activity: the reading of a science fiction short story connected to the Battle's theme. We think the short stories may help making Battles more scaffolded and replicable in other contexts though, as of the time of writing, we have not explored the deeper implications of the introduction of this extra step, and neither have we obtained empirical validation.

The immediate next step of this work will revisit the Battles' design and include the reading of the short story. We will also need to gather how the students' perceptions changed, exploring how short stories influence the students' understanding, their preparation for the debate, and the development of non-technical skills. Since many years, we always gathered students' feedback connected to Battles, which by now forms a small source of longitudinal data. This intervention, however, is likely to require an *ad-hoc* validation of the change, defining some new indicators for gauging its effects.

Another line of inquiry is connected to the selection of the short stories: should they remain the same across the years, or should they change? In this edition of the course, we pre-selected short stories from our own reading experience. If stories could change over the years, though, should we create a systematic selection procedure, for example using stories from collections that won literary prizes? Another alternative could be instead to base our selection on literature discussing reading science fiction as a curricular activity. Could we, for example, imagine a partnership between our I&E Basics course and courses such as Speculating Futures, where the latter proposes short stories and commentary, and the former uses them for debating? And, if so, how could we contribute back?

[1]See http://speculatingfutures.club/ or https://github.com/frnsys/speculating_futures (both URLs accessed 2023/10/11).

ACKNOWLEDGMENTS

The authors wish to thank all the colleagues that strongly contributed to the evolution and development of Tech Battles in these years, namely Matteo Bonifacio, Andrea Capaccioli, Andrea Guarise, Massimiliano Luca, Chiara Grossi, and Maurizio Marchese.

REFERENCES

[1] K.L. Alford and J.R. Surdu. 2002. Using In-Class Debates as a Teaching Tool. In *32nd Annual Frontiers in Education*. IEEE, Boston, MA, USA.
[2] Lorenzo Angeli, Massimiliano Luca, Chiara Grossi, Francesca Fiore, Andrea Capaccioli, Andrea Guarise, Milena Stoycheva, and Maurizio Marchese. 2020. Prove Me Wrong! How Debating Becomes the Secret Weapon to Teach ICT Students Innovation and Entrepreneurship. In *2020 IEEE TALE*. IEEE.
[3] Matteo Bonifacio, Lorenzo Angeli, and Milena Stoycheva. 2017. ENACTING DIVERGENT LEARNING DYNAMICS IN TEAMWORKING: THE CASE OF TECHNOLOGY BATTLES. In *EDULEARN 2017*. IATED.
[4] CC2020 Task Force. 2020. *Computing Curricula 2020: Paradigms for Global Computing Education*. ACM, New York, NY, USA. https://doi.org/10.1145/3467967
[5] A.M.S. Hamouda and F. Tarlochan. 2015. Engaging Engineering Students in Active Learning and Critical Thinking through Class Debates. *Procedia - Social and Behavioral Sciences* 191 (June 2015), 990–995.
[6] Donna J Haraway. 2016. *Staying with the trouble: Making kin in the Chthulucene*. Duke University Press.
[7] Ruth L. Healey. 2012. The Power of Debate: Reflections on the Potential of Debates for Engaging Students in Critical Thinking about Controversial Geographical Topics. *Journal of Geography in Higher Education* 36, 2 (May 2012), 239–257.
[8] Bruno Latour. 1987. *Science in action: How to follow scientists and engineers through society*. Harvard university press.
[9] Leah Zaidi. 2019. Worldbuilding in Science Fiction, Foresight and Design. *Journal of Futures Studies* 23, 4 (June 2019). https://doi.org/10.6531/JFS.201906_23(4).0003
[10] María L. Pertegal-Felices, Maria J. Manresa-Inigo, Jose L. Sanchez-Romero, and Antonio Jimeno-Morenilla. 2008. The Virtual Debate as a Complementary Evaluation and Learning Method for Large Groups: An Experience with First Year Computer Studies Students. In *8th IEEE International Conference on Advanced Learning Technologies*. IEEE, Santander, Cantabria, 900–904.
[11] Pezhman Zare and Moomala Othman. 2015. Students' Perceptions toward Using Classroom Debate to Develop Critical Thinking and Oral Communication Ability. *Asian Social Science* 11, 9 (April 2015), p158.

Evaluating Storytelling Videos Using YouTube Analytics

Anna Ly
annahuynh.ly@mail.utoronto.ca
University of Toronto Mississauga
Mississauga, Ontario, Canada

Tingting Zhu
tingting.zhu@utoronto.ca
University of Toronto Mississauga
Mississauga, Ontario, Canada

Andrew Petersen
andrew.petersen@utoronto.ca
University of Toronto Mississauga
Mississauga, Ontario, Canada

ABSTRACT

Prior research has shown that storytelling is an effective method for increasing comprehension of concepts. Students often find computational topics, such as data structures, to be difficult to grasp initially. To bridge this gap, we investigate whether the use of storytelling in pre-lecture videos increases students' retention and understanding. At a North American university, instructors randomly assigned students to two separate groups who watch different types of pre-lecture videos: one in a traditional format and the other where they teach a concept through storytelling. These videos were deployed as unlisted YouTube links embedded in students' quizzes. Using YouTube's Reporting API, we analyzed the audience retention data against elapsed time to compare audience retention between traditional and storytelling teaching methodologies. There were more storytelling videos with a higher average retention level, and the audience displayed less skipping behaviour than their traditional counterparts. In the future we will further analyze students' perceptions of storytelling videos to better understand higher audience retention and the effectiveness of learning through storytelling lectures.

ACM Reference Format:
Anna Ly, Tingting Zhu, and Andrew Petersen. 2024. Evaluating Storytelling Videos Using YouTube Analytics. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635503

1 INTRODUCTION

Practice problems are key to obtaining proficiency in computer science rather than passive lecture consumption. At our university, we use a flipped-classroom approach in computer science courses, which entails students watching pre-recorded videos before attending in-person lectures. During class sessions, students solve practice problems and have direct interactions with instructors, and this method has shown to increase student's learning [2].

Nevertheless, students struggle to understand concepts presented in the pre-lecture videos. Analytical insights from videos provide insights where students often encounter difficulties or need to review for full comprehension. We take inspiration from prior work demonstrating that storytelling can enhance student learning in various disciplines and has been effective at teaching adults [1].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635503

2 METHODOLOGY

Oppl et al. [3] proposed a method for in-depth analysis of data from educational videos. They recommended considering audience retention data, which represents the ratio of viewers watching a specific point of the video. It can be above 100% if watchers frequently replays portions of the video. This data is accessible on YouTube by using the Analytics & Reporting APIs, where the lecture videos were deployed.

Between January 2023 and May 2023, we deployed two different types of pre-lecture videos (storytelling and traditional) in a second year data structures course at a research-intensive North American university. During this semester, the course featured two lecture sections hosted by two different instructors. To minimize instructor bias, the instructors randomly assigned 50% of their students to groups A (n = 157) and B (n = 156).

Classes at this North American university run for 12 weeks. The two different types of pre-lecture videos were deployed in weeks 1, 6, 9, and 11. Group A were exposed to traditional pre-lecture videos for weeks 1 and 9, but watched storytelling videos for weeks 6 and 11. Group B watched the opposite. These videos were deployed through an unlisted YouTube link embedded in the quizzes, and students only had access to one type of videos. We designated the group that watched the traditional videos as the control group for the week, and the treatment group watched the storytelling edition.

We gathered data before (1) the pre-lecture quiz, (2) the relevant homework problems, (3) the midterm, and (4) the final exam. Week 1 had four pre-lecture videos, week 6 had 1 pre-lecture video, week 9 had 3 pre-lecture videos, and week 11 had 1 pre-lecture video. Following Oppl et al. [3]'s guide for analyzing YouTube data, we calculated the average retention level, the slope of the trend (represented by β from the simple linear regression of audience retention against elapsed time), and the linearity of the trend (represented by the coefficient of determination, R^2, which informs how well the simple linear regression model predicts audience retention.)

3 ANALYSIS & RESULTS

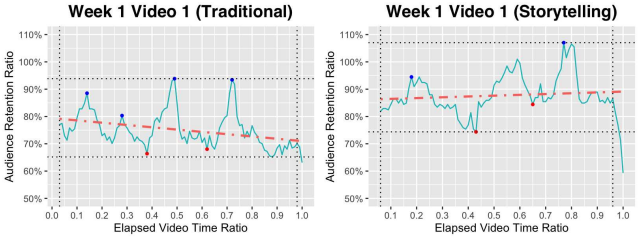


Figure 1: The audience retention of storytelling and traditional videos for week 1, pre-lecture video 1.

Table 1: Comparison of the different metrics against the traditional and storytelling videos prior to lecture. The higher average retention level is bolded for emphasis.

		Control (Traditional)					Treatment (Storytelling)				
		VL[a]	Views	ARL[b]	SOT[c]	LOT[d]	VL[a]	Views	ARL[b]	SOT[c]	LOT[d]
Week 1	Video 1	4:45	244	75.07459%	-0.08403	13.18%	4:06	199	**87.71663%**	0.02965	1.144%
	Video 2	6:33	201	**83.17347%**	-0.29016	43.64%	5:37	181	81.30904%	-0.34994	74.9%
	Video 3	8:59	250	81.54732%	0.27140	48.93%	8:04	208	**103.3464%**	0.51838	59.02%
	Video 4	7:35	54	58.75802%	-0.37416	68.73%	6:43	48	**78.72421%**	-0.06561	3.66%
Week 6	Video 1	9:01	168	**99.8467%**	0.39587	34.49%	8:35	163	81.53794%	-0.03902	1.48%
Week 9	Video 1	4:49	143	84.57194%	-0.08982	9.305%	5:41	132	**90.07408%**	-0.026683	3.171%
	Video 2	5:33	132	84.90258%	-0.28926	54.53%	8:34	122	**85.97713%**	-0.32981	63.07%
	Video 3	0:57	114	**121.0979%**	-0.34263	59.43%	2:48	101	97.75187%	-0.25740	43.19%
Week 11	Video 1	12:15	168	102.7878%	-0.39481	29.88%	12:08	159	**104.319%**	-0.60497	42.65%

[a] Video Length, [b] Average retention level, [c] Slope of trend (β), [d] Linearity of trend (R^2).

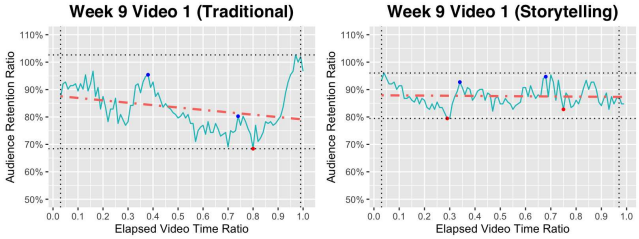


Figure 2: Similar to the first figure, but for week 9, video 1.

Using the YouTube Reporting API on Python, we were able to extract video metrics such as the number of views for a specific period and audience retention. Although we collected the YouTube audience retention throughout multiple time periods within the course, the number of views during the midterm and exam periods were sparse. Some videos' view count increased during the homework period, however there was no homework for week 11. Due to space limitations and to maintain consistency, we present the pre-lecture results for now. Table 1 shows a summary of collected video analytics.

As expected, most of the storytelling videos had a higher average retention level (for 6 out of 9 videos), meaning students watched a higher percentage of the video once the video starts. However, there were still three cases where the traditional videos have a higher retention level. On one (week 9), we believe that the short length and the lack of key frames explain the much higher retention rate. In another (week 6), the traditional video showed a big bump in retention (to 130%) around animations. Typically, videos will have a decrease of retention over time, which is represented by a negative slope. This is true for most of the storytelling and traditional videos.

Another way to observe student's viewing behaviours is to examine R^2 and analyze the presence of bumps and dips. We have attached pre-lecture videos in Figure 1 and in Figure 2 to emphasize said differences between retention levels. The blue dots denote local maximums, while the red dots denote local minimums. For week 1 video 1, the traditional video has prevalent skipping patterns through frequent bumps and dips. Retention rose when new lecture slides appeared. The storytelling video, which uses animations over classical lecture slides, has a less readily apparent skipping behaviour.

4 CONCLUSIONS AND FUTURE WORK

In this poster we compared the YouTube retention data between storytelling and traditional videos. We found that storytelling videos generally had a higher average audience retention level, indicating that students were more engaged with these videos compared to their traditional counterparts. Furthermore, repeated patterns of bumps and dips were more prevalent in traditional videos, where students notable increased their retention during key frames where new slides or text were introduced. Conversely, the storytelling videos contain less of a discernible pattern.

Although audience retention data can be beneficial for instructors, there are multiple interpretations which make it difficult to conclude whether storytelling videos have a beneficial impact on students' understanding. For example, higher retention could mean that a concept or explanation was more difficult to grasp so the audience needs to re-watch multiple times. From a preliminary reading of the survey responses, we believe our audience concluded that the videos are worth watching. In our future work, we will investigate further to understand why audience retention is generally higher on storytelling videos. Specifically, we will analyze time spent on tasks, reported attitudes towards different types of videos, performance, and factors that makes a video intervention more appealing to a particular cohort of students.

REFERENCES

[1] Enzo Caminotti and Jeremy Gray. 2012. The effectiveness of storytelling on adult learning. *Journal of Workplace Learning* 24, 6 (2012), 430–438. https://doi.org/10.1108/13665621211250333

[2] Michail N Giannakos, John Krogstie, and Nikos Chrisochoides. 2014. Reviewing the flipped classroom research: reflections for computer science education. In *Proceedings of the computer science education research conference*. 23–29. https://doi.org/10.1145/2691352.2691354

[3] Stefan Oppl, Anja Gutmann, Ivan Lazic, and Florian Mühlburger. 2019. Examining audience retention in educational videos-potential and method. In *2019 18th International Conference on Information Technology Based Higher Education and Training (ITHET)*. IEEE, 1–10. https://doi.org/10.1109/ITHET46829.2019.8937364

Computer Science Curriculum Trends

Sean Mackay
Department of Engineering Education
University at Buffalo
Buffalo, NY, USA
snmackay@buffalo.edu

Adrienne Decker
Department of Engineering Education
University at Buffalo
Buffalo, NY, USA
adrienne@buffalo.edu

ABSTRACT

Since the 1960s, the ACM has provided routinely updated guidelines for what concepts constitute a computer science curriculum, with the latest version currently in development in 2023. These guidelines have traditionally provided a model curriculum from which universities can choose to adopt or modify for their own purposes. What is unclear, however, is to what degree schools follow the curriculum recommendations that the ACM provides. While most faculty and students likely have knowledge of their own institution's curriculum, as well as what courses are offered at a small selection of other schools, the goal of the work presented in this poster is to distill a cohesive view of what computer science curriculums in their second and third years look like across a broad range of universities across a range of institutions. Our goal with this work was to answer the following question: *What do computer science course requirements look like at a wide range of different institutions?* We believe the work will help those who are trying to develop curriculum changes within their own institutions and aims to provide a more cohesive view of what trends and patterns exist in course offerings and degree requirements for computer science in the second and third years across a wide range of universities.

CCS CONCEPTS

•Social and professional topics~Professional topics~Computing education~Computing education programs~Computer Science Education

KEYWORDS

Computer Science Curriculum, curriculum review, curriculum recommendations

ACM Reference format:

Sean Mackay and Adrienne Decker. 2024. Computer Science Curriculum Trends. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V.2 (SIGCSE 2024). ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/3626253.3635501

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635501

1 INTRODUCTION

Computer science is a subject with a broad range of course requirements, subfields, and topics. What courses constitute a computer science degree can vary broadly across different universities and colleges. The Association for Computing Machinery (ACM) has routinely provided updates to their computing curriculum standards for computer science as well as a range of other computing-related disciplines since the 1960s [7]. The latest recommendation report for computer science is currently in development in 2023 and provides a broad range of changes to the curriculum, with emphasis for an increase in focus on human factors, ethics, and human-computer interaction [6]. While these recommendation reports are detailed and provide a wide array of valuable material for faculty at institutions to implement their own curriculum, it is unclear how well adopted these recommendations are.

The research presented in this poster aims to develop a more cohesive understanding for what computer science curriculums look like across a broad range of universities, including different styles and sizes of school, different geographic locations, and different rankings. Our goal was to answer the following research question: What do computer science course requirements look like at a wide range of different institutions? Our motivation for this work stemmed from research into second- and third-year students. When conducting this work, we realized that while the ACM recommendation report does provide guidelines, we were unsure how broadly these course recommendations are adopted across different types of institutions. To develop a better understanding for computer science curriculums, we conducted a study investigating curriculums across a broad range of universities both in the United States as well as several universities in Europe and Canada. Our goal with this work and this poster is to provide a broader sense of what curriculums look like at a range of institutions, allowing those who are currently working with their universities in curriculum develop a sense of what offerings exist at a range of schools.

2 BACKGROUND

There have been several reviews of curriculum trends across universities, though generally limited to the scope of a specific context or group of universities. In 2022, a curriculum review of liberal arts computing programs was published detailing several findings about what computer science curriculum in these schools entails [2]. The work was conducted from the context of

working towards developing the CS202x curriculum revisions now known as the 2023 ACM curriculum recommendation report for computer science. There have also been many publications focused on presenting curricular innovations piloted at different singular institutions [1, 3, 4]. However, these publications do not provide a cohesive view of curriculums from a range of different institutions.

3 METHODS

The work presented in this poster primarily consists of a records review of different university computer science curriculums based on what curriculum information is publicly available from each school. A total of 50 schools were initially included in this report from a broad range of different schools. US News and World Report provides a range of rankings for different schools in different categories. The following groups of schools were included in this curriculum review:

1. National Liberal Arts Colleges in the United States
2. Global Universities
3. National Universities in the United States
4. Regional Universities in the Northeastern United States

For each of these categories, we included the top five institutions according to US News and World Report's ranking, except for National Universities for which we included 10. Additionally, for each category, we included for review an equal number of randomly sampled universities at a range of different rankings. To accomplish this, we sorted each list of institutions alphabetically and randomly selected schools that were not already in the sample set.

Once we collected our list of institutions, we visited each school's publicly facing site. At this point, several universities were excluded from this study, as despite US News and World Report reporting these schools as offering degrees in computer science, the schools did not list these degrees as being offered. We suspect this is due to US News and World Report grouping computer science and information technology as being the same degree. In total, 29 institutions were included in this analysis.

For each school included in our sample, we reviewed each school's publicly available curriculum for computer science. Most schools provided a list of required courses as well as electives as well as course descriptions. For each curriculum, we coded each course as corresponding to each of the ACM curriculum recommendations knowledge areas. For courses without descriptions on departmental pages, individual course websites were sought to determine course content.

4 FINDINGS

Several patterns have emerged from this curriculum review. Firstly, there exist a set of core topics that are consistently required across most universities and colleges in this study, including:

1. Data Structures
2. Algorithms
3. Artificial Intelligence and Machine Learning
4. Software Engineering
5. Systems Programming

Notably, curriculums can be categorized into schools with very detailed and specific course requirements, and schools that provide students with a range of options from which to choose from. There is no perceivable pattern to this, with some liberal arts colleges having quite strict course requirements, while some national universities have very flexible requirements for their computer science degrees.

Trends can be observed in terms of schools having different research foci that become apparent in the range of course offerings and their curriculum requirements. Johns Hopkins University, for example, has a notable emphasis in their curriculum on medical computing such as computer assisted surgery appearing as a systems course requirement [5]. Other schools have a clear emphasis on artificial intelligence, programming languages, or distributed computing.

5 FUTURE WORK

The research presented in this poster exists as a portion of a larger body of research being conducted to investigate student learning during their second and third years of university education, particularly at concepts students find to be the most troublesome to learn during these years.

ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. National Science Foundation under Grant No. 2044179.

REFERENCES

[1] Wasim A. Al-Hamdani and Ivory J. Griskell. 2005. A proposed curriculum of cryptography courses. In Proceedings of the 2nd annual conference on Information security curriculum development (InfoSecCD '05), Association for Computing Machinery, New York, NY, USA, 4–11. DOI:https://doi.org/10.1145/1107622.1107625

[2] Jakob Barnard, Grant Braught, Janet Davis, Amanda Holland-Minkley, David Reed, Karl Schmitt, Andrea Tartaro, and James Teresco. 2022. Reflective Curriculum Review for Liberal Arts Computing Programs. J. Comput. Sci. Coll. 38, 4 (November 2022), 58–60.

[3] Bradley Bogolea and Kay Wijekumar. 2004. Information security curriculum creation: a case study. In Proceedings of the 1st annual conference on Information security curriculum development (InfoSecCD '04), Association for Computing Machinery, New York, NY, USA, 59–65. DOI:https://doi.org/10.1145/1059524.1059537

[4] Mikhail Fominykh, Fridolin Wild, Ralf Klamma, Mark Billinghurst, Lisandra S. Costiner, Andrey Karsakov, Eleni Mangina, Judith Molka-Danielsen, Ian Pollock, Marius Preda, and Aljosa Smolic. 2020. Model Augmented Reality Curriculum. In Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '20), Association for Computing Machinery, New York, NY, USA, 131–149. DOI:https://doi.org/10.1145/3437800.3439205

[5] Computer Science, Bachelor of Science < Johns Hopkins University. Retrieved October 11, 2023 from https://e-catalogue.jhu.edu/engineering/full-time-residential-programs/degree-programs/computer-science/

[6] CS2023 Version Gamma – CS2023. Retrieved October 11, 2023 from https://csed.acm.org/cs2023-gamma/

[7] Curricula Recommendations. Retrieved October 11, 2023 from https://www.acm.org/education/curricula-recommendations

Developing Interactive Exercise Materials for Machine Learning Using Spreadsheets

Atsuhiko Maeda
Kanagawa Institute of Technology
Kanagawa, Japan
maeda@cco.kanagawa-it.ac.jp

ABSTRACT

This paper presents a new technique for creating machine-learning exercise materials using spreadsheets. Spreadsheets are often used in teaching machine learning for beginners and non-specialists. However, computation in machine learning models is divided into a learning phase and an inference phase, and conventional spreadsheet-based materials either rely on the software's extensions for the learning phase or can result in huge sheets, which is unsuitable for learners who want to observe it. We realize the learning phase on a spreadsheet that fits within a PC screen, including visualization for better understanding, without using any extensions. Also, this implementation technique makes it possible to execute or pause the learning process interactively. We will show an example of a neural network implementation and discuss the merits and limitations of this technique.

ACM Reference Format:
Atsuhiko Maeda. 2024. Developing Interactive Exercise Materials for Machine Learning Using Spreadsheets. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635497

1 INTRODUCTION

As machine learning, exemplified by deep learning, permeates our daily lives, the need for literacy in this field is becoming increasingly relevant. This is important not only for future professionals in the area but also for individuals who don't become computer scientists or machine learning engineers. Curricula have been also developed to address this trend for students in non-IT fields[1, 6, 8].

For this purpose, spreadsheet software is one of the most effective platforms to help learners understand how machine learning works. Many educational materials on machine learning using spreadsheets have been proposed[2–5, 7].The advantage of using a spreadsheet is that it is usually easier to learn than programming languages and more straightforward to study and explore functional relationships among multiple parameters of a machine learning model[3]. However, the computational process in machine learning models can be divided into a learning phase, in which models are trained, and an inference phase, in which the models are used, and most of the exercise materials using spreadsheets so far either rely on the software's extensions for the learning phase or require huge sheets, making it challenging to observe the phase.

This paper proposes a technique to realize machine learning exercise materials, including implementation of the learning phase and the visualization for better understanding, on a spreadsheet that fits on a PC screen in an easy-to-read manner without using any software extensions. We will show an example of a neural network implementation and discuss the merits and limitations of this technique.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635497

2 RELATED WORK

Early machine learning exercise materials using spreadsheets targeted only the inference phase[3], but later, two types of materials were proposed to support the learning phase[2, 4, 5, 7].One is to implement only the inference phase and rely on the solver add-ins for mathematical optimization for the learning phase[2, 5]. In this method, the latter phase is opaque to the learners.The other method is the "append method." For machine learning, it is necessary to update the model's parameter values iteratively. However, updating the values stored in a spreadsheet's cells is difficult due to the software's structure. In Microsoft Excel, when a parameter update formula like $w = w - \alpha \frac{\partial E}{\partial w}$ is entered into a cell that should store the updated value of a parameter, a circular reference error occurs if the formula contains the reference to that cell even indirectly. Therefore, the conventional solution is to append the updated values and the intermediate results to new cells for each iteration of learning[4, 7], which can result in a spreadsheet many times larger than the PC screen as the number of iterations increases, and this reduces its readability.

3 "OVERWRITE METHOD" TO REALIZE THE LEARNING PHASE

We propose a new method called the "overwrite method" to realize the learning phase. In this method, once the training is iterated once and the updated values of the parameters are output to certain cells, have the user manually copy and paste them as "values" not as the default "formulas" into the cells that contain the initial values of the parameters. This avoids circular references. However, repeating this sequence of operations is tedious. Thus, we suggest that after completing the above operations only once, the user should repeat the sequence by holding down the F4 key, a shortcut to Excel's auto-repeat function. This also makes it possible to run or pause the learning process interactively.

4 EXAMPLE OF NEURAL NETWORK

We show an example of applying our implementation technique to a 3-layer neural network performing binary classification (Figure

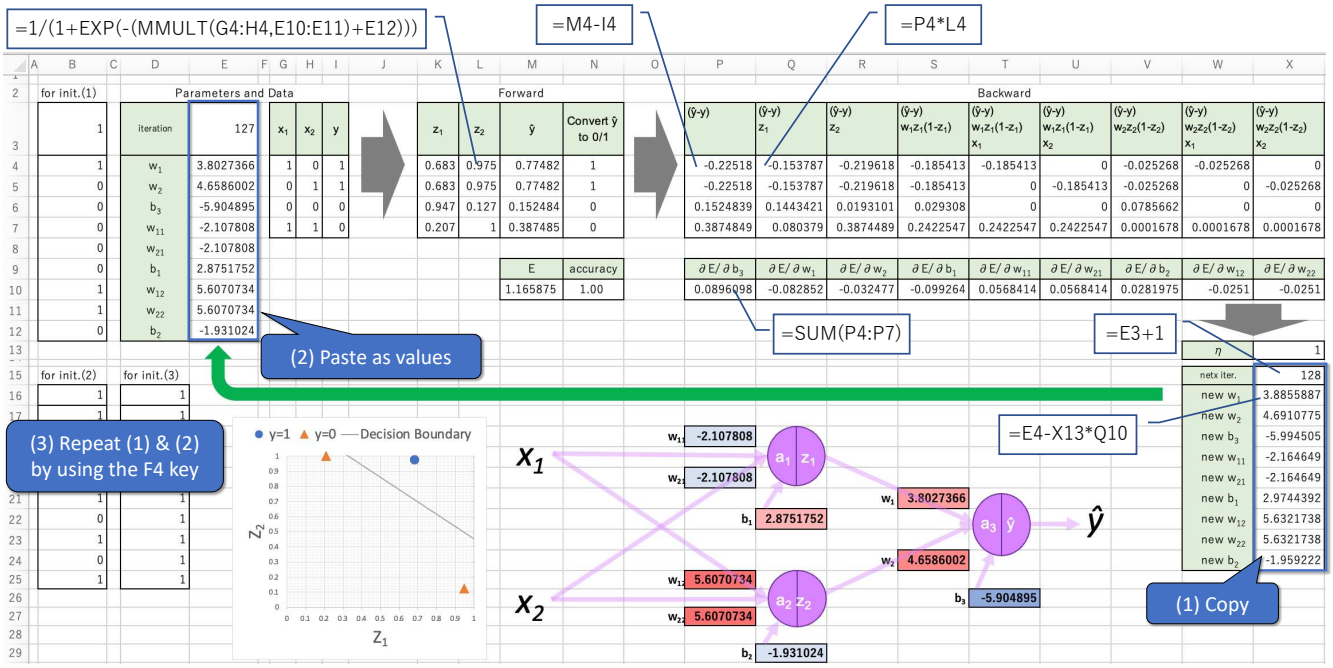


Figure 1: Proposed implementation of a neural network with backpropagation and its operation flow.

1). Cells E4 to E12 store the initial values of the model's parameters. The model is illustrated below, referring to these cells, with the magnitude of the parameter indicated by the color. In the graph below, we can observe that the decision boundary moves during learning, and the coordinates showing the output values of the intermediate layer also move so that they can be separated at the boundary. Cells G4 to I7 contain an XOR truth table as training data. The backpropagation is used as the learning method. This algorithm performs forward propagation, meaning normal inference, first. This is implemented in cells K4 through N7. Next, the partial derivatives of error function E with respect to model parameters are performed. The error function measures how far the model outputs are from the correct answers, and here the binary cross-entropy is applied. The derivatives are calculated using the outputs of forward, in order from the output layer to the input layer, which is called backward propagation and is implemented in Cells P4 through X10. The vector built from the derivatives is called the gradient, represented in Cells P10 to X10. Then, the values of the parameters are updated with Cells X16 to X24 using the gradient.

To train the model, the user copies Cells X14 to X22 containing the updated values of the parameters, pastes them into Cells E3 to E12 as values, and then repeats these operations using the F4 key. In this example, learning is accomplished in a few hundred iterations or tens of seconds, but the user can also pause or incrementally iterate the learning process.

5 DISCUSSION

We intend to lecture students on machine learning algorithms and then have them use this material in hands-on exercises to implement what they have learned and to test the implementation. One way to practice implementation would be to have the students fill in some of the cells in this material by leaving them blank. In this situation, the fact that all the implementation fits within the screen would help promote understanding. In training the neural network model, the novice learner should be allowed to see that it can classify linearly inseparable data, such as XOR, unlike logistic regression, etc. The visualization and the functionality to pause or incrementally iterate the training would be helpful. On the other hand, the proposed technique is unsuitable for creating examples that use many parameters or training data. However, even with the number of data and parameters as in this implementation example, it would be possible to observe how the neural network is learning.

REFERENCES

[1] Monique Brouillette. 2019. AI added to the curriculum for doctors-to-be. *Nature Medicine* 25 (2019), 1808–1809. https://doi.org/10.1038/s41591-019-0648-3

[2] Tarek Hegazy and Amr Ayed. 1998. Neural Network Model for Parametric Cost Estimation of Highway Projects. *Journal of Construction Engineering and Management* 124, 3 (1998). https://doi.org/10.1061/(ASCE)0733-9364(1998)124:3(210)

[3] Thomas T. Hewett. 1985. Teaching students to model neural circuits and neural networks using an electronic spreadsheet simulator. *Behavior Research Methods, Instruments, & Computers* 17 (1985), 339–344. https://doi.org/10.3758/BF03214406

[4] Thomas F. Rienzo and Kuriakose K. Athappilly. 2012. Introducing Artificial Neural Networks through a Spreadsheet Model. *Decision Sciences Journal of Innovative Education* 10, 4 (2012), 515–520. https://doi.org/10.1111/j.1540-4609.2012.00363.x

[5] Serhiy Semerikov et al. 2020. Using spreadsheets as learning tools for computer simulation of neural networks. In *SHS Web of Conferences*, Vol. 75. https://doi.org/10.1051/shsconf/20207504018

[6] Anu Vazhayil et al. 2019. Focusing on Teacher Education to Introduce AI in Schools: Perspectives and Illustrative Findings. In *2019 IEEE Tenth International Conference on Technology for Education (T4E)*. 71–77. https://doi.org/10.1109/T4E.2019.00021

[7] Yoshiyuki Wakui and Sadami Wakui. 2019. *Excel de wakaru Kikai Gakusyū Chōnyūmon (in Japanese)*. Gizyutsu Hyōronsya.

[8] Jennifer J. Xu and Tamara Babaian. 2021. Artificial intelligence in business curriculum: The pedagogy and learning outcomes. *International Journal of Management in Education* 19, 3 (2021). https://doi.org/10.1016/j.ijme.2021.100550

A Generalized Framework for Describing Question Randomization

Romina Mahinpei*
mahinpei@student.ubc.ca
Computer Science
The University of British Columbia
Vancouver, Canada

Iris Xu*
irisxu02@student.ubc.ca
Computer Science
The University of British Columbia
Vancouver, Canada

Steven Wolfman
wolf@cs.ubc.ca
Computer Science
The University of British Columbia
Vancouver, Canada

Firas Moosvi
firas.moosvi@ubc.ca
Computer Science
The University of British Columbia
Vancouver, Canada

ABSTRACT

The rise of online assessments has motivated the development of randomized question banks, with randomization referring to the generation of different variants of a question. Although not all randomization efforts are equally effective in generating question isomorphs, the current classification of questions solely as randomized or not fails to address the varying degrees of randomization. To address this limitation in describing the diversity of randomization designs, we introduce a framework that outlines six distinct randomization levels. Additionally, we designed practical guides to assist educators in effectively using the framework, aligning with their pedagogical objectives. Through our application of this framework to classify around 200 questions from two courses, we further highlight the generalizability of the framework and reveal insights into the considerations and challenges associated with incorporating question randomization into computer science curricula.

CCS CONCEPTS

• **Applied computing** → **Education**.

KEYWORDS

Question randomization, Online assessments, Question design

ACM Reference Format:
Romina Mahinpei, Iris Xu, Steven Wolfman, and Firas Moosvi. 2024. A Generalized Framework for Describing Question Randomization. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635599

*Both authors contributed equally to this research.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635599

1 INTRODUCTION

Randomizing questions is a strategy used by educators to generate question variants within online assessments to enhance the quality of instruction. The recent shift toward online instruction and asynchronous online assessments has increased both the need and adoption of question randomization. The rising interest in alternative grading systems and mastery learning gives students multiple opportunities to demonstrate proficiency. For that, autograded randomized question banks are needed. While previous studies have investigated the effectiveness of randomizing questions to prevent cheating [1, 2], there exists a notable gap in describing the design and implementation of randomization in questions. Randomization can vary in extent and magnitude and potentially yield different levels of efficacy in promoting learning or improving exam security. To address this gap, we propose a six-level question randomization framework that guides educators in systematically implementing and evaluating randomization within questions.

2 RANDOMIZATION FRAMEWORK

The proposed randomization framework is provided in Table 1. The current version classifies randomization into six levels ranging from 0 to 5. Importantly, levels in this framework are not necessarily on a quality hierarchy from worst to best. A question often belongs to multiple levels. This scenario is possible due to the non-conflicting nature of the implementation techniques for different randomization levels. Questions of relatively higher complexity, such as multi-part questions with many question parameters, may have a mix of level 1-4 randomization. Further, note that level 5 randomization mainly applies in the context of assessments since it requires entirely independent variants. As an example, this can be achieved by implementing a large question bank from which the assessment questions for each student are randomly chosen.

3 FRAMEWORK APPLICATION

The randomization framework was applied to online assessment questions from an algorithms and data structures course ($n = 86$) and an operating systems course ($n = 100$). The framework application was done by two students under the supervision of one faculty member who had drafted the original framework and one instructor who had taught both courses. This process started with four

Table 1: Question randomization framework. Levels 0-4 pertain to individual questions and level 5 (gray) applies to assessments. {{...}} represents a randomization point and its content can be replaced by randomly generated alternatives.

Level	Label	Description	Purpose	Example
0	Unrandomized	Every student receives precisely the same question.	Classify questions that are not randomized at all.	Abdallah uses a perfect binary tree to implement a small database system with 63 nodes. How many leaves will this tree have?
1	Surface Features	Surface level features (e.g., names, colours, phrases) change for each variant.	Increase cognitive load for students when they are pattern-matching.	{{Abdallah}} uses a perfect binary tree with 63 nodes to implement a {{small database system}}. How many leaves will this tree have?
2	Conditions	Within a single problem scenario, conditions and values change for each variant.	Offer opportunities for repeated retrieval practice and more productive group work.	Abdallah uses a perfect binary tree to implement a small database system with {{63}} nodes. How many leaves will this tree have?
3	Scenarios	Problem scenarios change for each variant, assessing a single concept.	Enable students to develop strategies, algorithms, and procedures to solve multiple problem scenarios.	Abdallah uses an {{unbalanced}} binary tree to implement a small database system with 63 nodes. What {{max height}} will this tree have?
4	Concepts	Randomized variations lead to an assessment of different concepts.	Diversify question style so students need to synthesize knowledge across concepts and cannot easily pattern-match.	Abdallah uses an {{M-ary tree with $m = 4$}} to implement a small database system with 63 nodes. How many leaves will this tree have?
5	Different Questions	Each question variant is entirely independent.	Provide students with authentic reassessment opportunities to demonstrate proficiency.	{{Abdallah uses a perfect binary tree to implement a small database system with 63 nodes. How many leaves will this tree have?}}

rounds of calibration each consisting of the reviewers classifying 10 questions independently before engaging in a group discussion to ensure consistency in classifications. After the calibrations, the student reviewers classified the remaining questions independently before discussing their ratings with the entire team. To assess inter-rater reliability (IRR) for the calibration and official rounds, Gwet's AC1 coefficient [3] was computed as a per-level IRR metric, and percentage agreement was computed as a per-question IRR metric. The AC1 coefficient was chosen as the IRR metric over other common scores such as Cohen's Kappa, since these common metrics underestimate IRR when there is good general agreement [3].

The first calibration round consisted of the most disagreement both at the aggregated per-question and at per-level bases, with a majority of the disagreement stemming from reviewers missing randomized features. Subsequent calibration rounds performed better than the first round and performed similarly to each other both at the aggregated per-question and per-level bases. For both the aggregated per-question and per-level IRR metrics, the official rounds performed similarly to the latest calibration. The two official rounds had an average percentage agreement of 94.4% and 96.1% respectively, with the lowest AC1 score being 0.74 (for Level 3) during the first round and the lowest AC1 score being 0.84 (for Level 2) during the second round. A notable source of disagreement that arose during the group discussion was the difficulty in differentiating scenarios from concepts. Since a distinct 'concept' in one course may be only one feature of a larger aggregated 'concept' in a more advanced course, making this distinction required all reviewers to be adequately familiar with the curriculum of the course from which questions are classified.

To address this continued presence of disagreement, additional guides to complement the randomization framework were created. This includes a decision tree to assist reviewers by outlining sequential steps and posing relevant questions to facilitate the systematic application of the framework, and a usage table to assist instructors by outlining several pedagogical and logistical considerations associated with each level of the framework.

4 CONTRIBUTIONS

The development and application of the six-level randomization framework offers a significant contribution to instruction and assessments, particularly in the context of computer-based questions. This novel framework establishes a standardized method for describing question randomization, allowing educators to design randomized questions to meet pedagogical objectives. Having a framework to systematically describe the current diversity in randomization also enables future research in the areas of randomization in testing and studies on the impact on learning and fairness. It serves as a valuable metric, helping describe the extent of randomization in a question, thereby enabling the benchmarking of study results related to randomization. It is also a versatile tool with implications extending beyond immediate instructional applications, facilitating ongoing research and advancements in CS education.

REFERENCES

[1] Binglin Chen, Matthew West, and Craig Zilles. 2018. How much randomization is needed to deter collaborative cheating on asynchronous exams?. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*. ACM, 1–10. https://doi.org/10.1145/3231644.3231664

[2] Samuel S. Chua, Joshuel B. Bondad, Zechariah R. Lumapas, and Joven D L. Garcia. 2019. Online Examination System with Cheating Prevention Using Question Bank Randomization and Tab Locking. In *2019 4th International Conference on Information Technology*. IEEE, 126–131. https://doi.org/10.1109/INCIT.2019.8912065

[3] Kilem Li Gwet. 2008. Computing inter-rater reliability and its variance in the presence of high agreement. *The British journal of mathematical and statistical psychology* (2008), 29–48. https://doi.org/10.1348/000711006X126600

Mastery with Method: Calibrating Policies to Boost Completion and Sentiment in a Computing Course using Mastery Learning

Vedansh Malhotra
UC Berkeley
Berkeley, CA, USA
vedansh@berkeley.edu

Jenny Mendez Mendez
UC Berkeley
Berkeley, CA, USA
jennybmendez@berkeley.edu

Daniel Garcia
UC Berkeley
Berkeley, CA, USA
ddgarcia@berkeley.edu

ABSTRACT

In our introductory computer science course at a large public university, we embarked on a transformative journey inspired by Grading for Equity [1, 3]. Our goal: to accommodate differential learning, allowing students to attain a fixed level of mastery, albeit over a variable timeframe.

Our course has six major project assessments that comprise 46% of the overall grade. In Fall 2022 (n=234), despite warnings [4],we allowed indefinite extensions on projects and observed a drastic downturn in project completion rates relative to previous terms.

The Spring 2023 iteration (n=107) aimed to rectify this. Students could acquire two-week (or a single emergency one month) extensions. Further extensions obligated a student-staff interaction during dedicated support hours. Completion rates and lateness improved compared to fall, but were still poor compared to historical data. We also observed heightened stress.

Summer 2023 (n=59) presented unique challenges: a smaller cohort, different demographics, and a halved timeframe. Nonetheless, we further refined our approach: 72-hour extensions were auto-approved, with additional time warranting a conversation with staff. Further, we mandated attendance to ensure students stay on-pace with material. The outcomes were stellar: near-perfect project completion rates, reduced lateness, and rejuvenated student sentiments. Fall 2023 (n=214) was our testbed for this strategy in a standard term, and early results showcase markedly better metrics compared to Fall 2022 and Spring 2023.

In conclusion, while embracing a flexible, mastery-learning approach is vital to accommodate differential learning, finding a judicious equilibrium between autonomy and structure can be pivotal in navigating academic outcomes and student well-being.

KEYWORDS

Grading for Equity, Procrastination, Differential Learning

ACM Reference Format:
Vedansh Malhotra, Jenny Mendez Mendez, and Daniel Garcia. 2024. Mastery with Method: Calibrating Policies to Boost Completion and Sentiment in a Computing Course using Mastery Learning. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635629

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635629

1 BACKGROUND

Our path towards a mastery learning model began in Spring 2019, at the *Grading for Equity* workshops which fundamentally reshaped our views on lateness and learning [2]. We realized that our "one-size-fits-all" model offered no flexibility to students who needed more time to master the material. Despite our encouraging project completion *rates*, which have historically hovered around 90%, our average project *scores* would follow a traditional bell curve; collectively, the staff were not satisfied with this status quo.

As such, we began the process of refactoring our grading policies and project assessments to support a mastery learning model at scale. We were already grading using absolute grading, rather than curved grading, and felt no change was needed there. Previously, students used to have three "slip day" tokens, allowing a single day's extension for project deadlines. After the tokens were exhausted, each subsequent day of delay incurred a fixed deduction — this was a department-wide norm, since most of the grading was done by *human* graders who needed to have a fairly consistent workload.

This framework saw a radical transformation with our pivot to automated grading. Now, students could, in theory, submit all projects by the term's end without overwhelming human graders. This shift to automation was pivotal — beyond alleviating grading burdens, it allowed us to reallocate significant time resources: our grading time was effectively cut in half, and we channeled these newfound staff hours into offering additional office hours, providing students more opportunities for individualized assistance.

2 ITERATIVE REFINEMENT

In Fall 2022, we announced our new model, the cornerstone of which was the assertion that "any points lost on ANY grading element can be earned back with no penalty if you keep on working." This announcement was met with palpable enthusiasm, exemplifying students' yearning for a more empathetic educational approach. All of our projects came with a suggested completion date. However, we emphatically conveyed that there were no lateness penalties, and that students who wished to submit past the deadline could do so after filling out a simple Google Form that would allow us to track their progress. Despite these well-intentioned measures, the fall semester painted a grim picture — the newfound flexibility had led to a significant dip in our project-completion rates, shown in Figure 1, with many students never starting to work. Two notable exclusions are our essay project, which is manually graded and can't admit resubmissions, and a final project which wasn't offered in Fall 2022 due to unforeseen disruptions.

In light of the fall semester's outcomes, Spring 2023 brought about an adjustment in our communication strategy. Our "no lateness penalties" approach, though still in play, was not advertised as

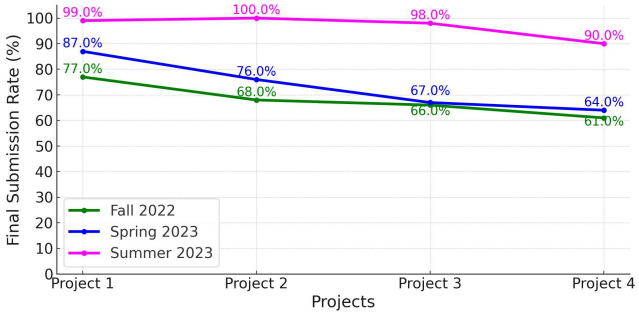


Figure 1: Final Completion Rates by Project and Term

emphatically. The messaging on the surface was stern, with the syllabus stating that "late submissions for all assignments in this course will not be considered, and will be awarded zero credit." However, we made it abundantly clear that submissions were only marked late if students did *not* request an extension *and* turned in work past the deadline. We implemented automatic two-week extensions on all projects, and a one-month emergency extension on any one project. Understanding that some students might need more time, we set up dedicated student-support office hours wherein students could speak with a staff-member to request an extension beyond the auto-approved threshold. While we did not explicitly note that all extensions will be auto-approved, our goal was *not* to mislead students — we made sure to periodically remind them that they should reach out if they felt overwhelmed and needed more time. Further, we often emailed students who fell behind on their submissions, offering a support meeting. This restructuring with added friction and support led to a rejuvenated commitment to deadlines, with the course witnessing a heartening surge in completion rates (though the metrics were still deficient compared to historical data.) Unfortunately, we also observed poorer student-sentiment in our feedback surveys, depicted in Figure 2, with many students noting that they experienced significant stress after falling behind.

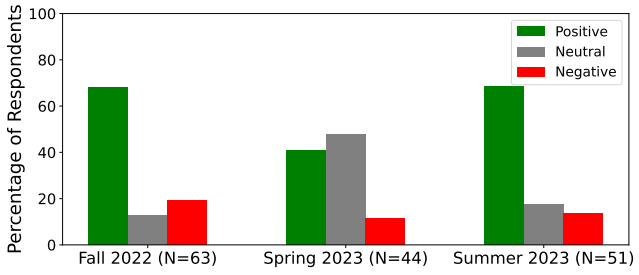


Figure 2: Student Feedback Sentiment Analysis by Term

Summer terms, traditionally compressed, necessitated agility in our policies and presented an opportunity to further fine-tune our approach. Unlike previous semesters, the announced policy was now that all late submissions would be penalized by a forfeiture of half the available credit. Extensions were limited to a 72-hour period per project and were auto-approved. For further extensions, students would have to schedule meetings with a staff member — as in spring, they were encouraged to ask as many times as needed, and requests were never denied. What further set the summer apart was the introduction of mandatory attendance in lectures and discussion sections. By ensuring that students consistently engaged with the material, we proactively set the course for them to stay on track. Students had the leeway to miss a handful of lectures and discussions without penalty, but further excused absences necessitated a conversation with the instructor — once again, these requests were never denied, but we didn't advertise this. Despite the seemingly harsher policies, our core ethos from Fall 2022 held firm: students were never penalized for late submissions or absences — it was the execution and messaging that had evolved. Indeed, summer culminated with near-perfect completion rates, reflecting a successful amalgamation of structured guidance and autonomy that resonated profoundly with our students.

However, it's essential to recognize that summer terms inherently differ from their fall and spring counterparts. The true litmus test of our new policies was set in Fall 2023, wherein we adopted the Summer 2023 policies without alteration. Encouragingly, preliminary data suggests a leap in completion rates. By the 7th week of instruction, the average completion rate for Project 1 reached an impressive 92%. Remarkably, this mid-term rate eclipses the end-of-term submission rates observed in both Fall 2022 (77%) and Spring 2023 (87%). Further, the submission rate for Project 2, standing at 73%, already surpassed the final rate from Fall 2022 (68%) and is on track to match (or surpass) Spring 2023's rate of 76%.

3 RECOMMENDATIONS

While our findings are rooted in preliminary research and an exhaustive analysis of Fall 2023 data remains on our agenda, our three-semester pilot has imparted invaluable lessons.

Broadly, we advocate for the integration of mastery learning in academic settings. The adoption process requires minimal adjustments: eradicating late penalties and transitioning to auto-grading. For educators eager to implement this paradigm, our advice is succinct: a grand proclamation of infinite flexibility in deadlines might inadvertently deter students from initiating their work. A more prudent approach involves subtler messaging about the availability of extensions. Set clear expectations for on-time submissions but reassure students that extensions stand as a supportive backup, available for ***anyone*** who needs it. Our Summer 2023 experience, validated by both student sentiment and performance metrics, underscores the efficacy of another pivotal policy: mandated attendance. This ensures consistent engagement and acts as a deterrent to academic lag, fostering an environment where fewer students trail behind.

REFERENCES

[1] Andrew Berns, J. Philip East, and J. Ben Schafer. 2021. *Grading for Equity: A Curriculum Development and Grading Process to Enhance Instruction.* Association for Computing Machinery, New York, NY, USA, 1351. https://doi.org/10.1145/3408877.3432503

[2] Benjamin S. Bloom. 1968. Learning for Mastery. Instruction and Curriculum. Regional Education Laboratory for the Carolina and Virginia, Topical Papers and Reprints, Number 1. *Evaluation comment* 1, 2 (1968), n2.

[3] J. Feldman. 2018. *Grading for Equity: What It Is, Why It Matters, and How It Can Transform Schools and Classrooms.* SAGE Publications. https://books.google.com/books?id=ORhoDwAAQBAJ

[4] Kristin Stephens-Martinez. 2022. Soft, hard, and late deadlines (or another failed experiment). https://ksm-cs.blogspot.com/2022/02/soft-hard-and-late-deadlines-or-another.html

Perception, Trust, Attitudes, and Models: Introducing Children to AI and Machine Learning with Five Software Exhibits

Fred Martin
fred.martin@utsa.edu
University of Texas at San Antonio
San Antonio, Texas, USA

Saniya Vahedian Movahed
saniya.vahedianmovahed@utsa.edu
University of Texas at San Antonio
San Antonio, Texas, USA

James Dimino
jamesdimino01@gmail.com
University of Massachusetts Lowell
Lowell, Massachusetts, USA

Andrew Farrell
andrew_farrell@student.uml.edu
University of Massachusetts Lowell
Lowell, Massachusetts, USA

Elyas Irankhah
elyas_irankhah@student.uml.edu
University of Massachusetts Lowell
Lowell, Massachusetts, USA

Srija Ghosh
srijaghosh2015@gmail.com
University of Massachusetts Lowell
Lowell, Massachusetts, USA

Garima Jain
garima_jain@student.uml.edu
University of Massachusetts Lowell
Lowell, Massachusetts, USA

Vaishali Mahipal
vaishali_mahipal@uml.edu
University of Massachusetts Lowell
Lowell, Massachusetts, USA

Pranathi Rayavaram
nagapranathi_rayavaram@student.uml.edu
University of Massachusetts Lowell
Lowell, Massachusetts, USA

Ismaila Temitayo Sanusi
ismaila.sanusi@uef.fi
University of Eastern Finland
Joensuu, Finland

Erika Salas
erika_salas@student.uml.edu
University of Massachusetts Lowell
Lowell, Massachusetts, USA

Kelilah Wolkowicz
kelilah_wolkowicz@uml.edu
University of Massachusetts Lowell
Lowell, Massachusetts, USA

Sashank Narain
sashank_narain@uml.edu
University of Massachusetts Lowell
Lowell, Massachusetts, USA

ABSTRACT

Artificial intelligence (AI) and machine learning (ML) have a deepening impact in our world. For empowered citizenship and career readiness, elementary and middle school students need to understand these technologies. This poster reports on five original interactive AI and ML software exhibits tested by 125 elementary and middle school students aged 7 to 14 years. Four themes emerged: Students recognized that AI and ML systems can process data from cameras (perception); they saw that these systems responded to their training input (trust); they appreciated the practical import of AI/ML systems (affective and cognitive attitudes); and students were introduced to models and modes (specialization).

ACM Reference Format:
Fred Martin, Saniya Vahedian Movahed, James Dimino, Andrew Farrell, Elyas Irankhah, Srija Ghosh, Garima Jain, Vaishali Mahipal, Pranathi Rayavaram, Ismaila Temitayo Sanusi, Erika Salas, Kelilah Wolkowicz, and Sashank Narain. 2024. Perception, Trust, Attitudes, and Models: Introducing Children to AI and Machine Learning with Five Software Exhibits. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635512

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635512

1. MOTIVATION We are just beginning to learn how to introduce children to ideas in artificial intelligence (AI) and machine learning (ML). Existing work includes constructionist environments like Machine Learning for Kids [4] and MIT App Inventor [5]; interactive tools such as Google's Teachable Machine [3]; and games such as Code.org's AI for Oceans [1] and Luden.io's while True: learn() [7].

Our work is situated in informal education: interactions which deliver a learning outcome in a short encounter. We are directly inspired by the work of Long, Blunt, and Magerko in creating AI literacy exhibit prototypes [2]. We build upon the "Five Big Ideas in AI" identified by the AI4K12.org project, which has framed thinking about K-12 artificial intelligence education worldwide [6].

2. BACKGROUND The first author offered a special topics course in Spring 2023 entitled "Developing AI Tools for Children." Ten students and two other faculty members participated in the course (the co-authors). The group met weekly to conceive and implement five original interactive software experiences introducing ideas in AI and ML.

To test our work, we collaborated with a public K–8 school partnered with our university to create a museum-like experience during the school day. In five class groups, a total of 125 children with ages ranging from 7 to 14 years old interacted with our exhibits over two days. Each group cycled around the exhibits, which

Table 1: Overview of the Five Exhibits

Exhibit	Introduction	AI4K12 Big Idea	Student Outcomes
AI for American Sign Language	Have computer recognize video hand gestures for letters of the ASL alphabet	1–Perception	How sample quality affect model performance
Tic-Tac-Toe	Play Tic-Tac-Toe against your choice of three different opponents	2–Representation & Reasoning	Recognizing AI has different capabilities; trying to beat the computer
Chemistry Safety Lab (ChemAIstry)	Specify to computer which objects are lab-safe and see results of training	3–Learning	Understand the significance of the training model
Ask Me Anything	Interact with chatbot with questions on dinosaurs, shoes, and astronomy	4–Natural Interaction	Insights into students' trust of AI
AI for Self-Driving Cars	See behavior of on-screen self-driving car using three different training models	5–Societal Impact	Understanding the potential impacts of inadequate training data

each had two university students or faculty members dedicated to helping the children. Facilitation and assistance was also provided by school teachers and staff. IRB permission to conduct the study and parental consent and student assent was obtained.

3. THE EXHIBITS The five exhibits touch on all of AI4K12.org's Five Big Ideas, with each project emphasizing one in particular: (1) AI for American Sign Language, where students' hand gestures were recognized by pre-trained models as individual letters of the ASL alphabet [perception]; (2) Tic-Tac-Toe, where students played against three personified computer opponents, each with a different algorithm/skill-level [representation and reasoning]; (3) Chemistry Safety Lab, where students provided training examples of safe and unsafe items and saw the results of their training as applied to other examples [learning]; (4) Ask Me Anything, a child-safe chatbot to have conversations on three topics of interest to children [natural interaction]; and (5) AI for Self-Driving Cars, a simulation of how autonomous cars perform when trained on different obstacles they will encounter in driving [societal impact]. Table 1 has a summary of the exhibits.

4. DISCUSSION Four key themes emerged:

1. Students recognized that AI and ML systems can receive and process data from cameras (perception). AI for ASL directly used webcams and AI for Self-Driving Cars which simulated the use of vision for "seeing" the obstacles on the car's roadway.

2. Students recognized that these systems provided useful information and responded to students' training input (trust). The exhibits responded directly to student input in predictable ways which built trust. In AI for ASL, students' hand gestures were recognized. In Tic-Tac-Toe, the system responded to their game moves. In ChemAIstry, students saw their training data used to make subsequent classifications. In AMA, the chat agent responded to with pertinent information. In AI for Self-Driving Cars, students saw how the selected model responded to obstacles present.

3. Students appreciated the practical value of the AI technologies (affective and cognitive attitudes). With AI for ASL, students realized they could learn to communicate with friends who were hard of hearing. In the Tic-Tac-Toe exhibit, students sought to challenge themselves, preferring the most-competent opponent, "Chris." With the AMA chatbot, students considered that it could be helpful in completing their homework.

4. Students were introduced to models and modes (specialization). AI for ASL had two models—Blue and Red—and students chose between them when performing letter gestures. In Tic-Tac-Toe, students selected among three different players (modes). In ChemAIstry, students performed explicit training of a classifier. This exhibit was unique in that students trained the model (vs. selecting among pre-existing models). With the AMA chatbot, students could select which topic they wished converse about with the chatbot (modes). In AI for Self-Driving Cars, three different models could be selected, each with different performance.

This work illustrates how AI tools can enrich children's learning and entertainment experiences in a variety of ways, while also highlighting the importance of tool-specific design and application considerations. Children's ability to interact and learn from AI tools suggests a promising future for integrating diverse AI tools into their lives.

ACKNOWLEDGMENTS This material is based upon work supported by the National Science Foundation under Grant IIS-2112633 (AI-CARING).

REFERENCES

[1] AI for Oceans. 2023. https://code.org/oceans
[2] Duri Long, Takeria Blunt, and Brian Magerko. 2021. Co-designing AI literacy exhibits for informal learning spaces. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–35.
[3] Teachable Machine. 2023. https://teachablemachine.withgoogle.com
[4] Machine Learning for Kids. 2023. machinelearningforkids.co.uk
[5] MIT App Inventor. 2023. https://appinventor.mit.edu
[6] David Touretzky, Christina Gardner-McCune, Fred Martin, and Deborah Seehorn. 2019. Envisioning AI for K-12: What should every child know about AI?. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 9795–9799.
[7] while True: learn(). 2023. https://luden.io/wtl

Impacts of Academic Preparedness on CS1 Performance

Barbara Martinez Neda
University of California, Irvine
Irvine, CA, US
barbarm@uci.edu

Flor Morales
University of California, Irvine
Irvine, CA, US
fvmorale@uci.edu

Kitana Carbajal Juarez
University of California, Irvine
Irvine, CA, US
kitanac@uci.edu

Jennifer Wong-Ma
University of California, Irvine
Irvine, CA, US
jwongma@uci.edu

Sergio Gago-Masague
University of California, Irvine
Irvine, CA, US
sgagomas@uci.edu

ABSTRACT

More than 50% of Computer Science (CS) students at the University of California, Irvine (UCI) experienced academic probation over a 10-year span. Particularly concerning, underrepresented groups (URG) faced an even higher probation rate, and probation students were twice as likely to leave the CS program.

We conducted a comprehensive survey involving 308 CS1 students at UCI to delve further into their past and present academic experiences. Specifically, we studied (1) the role of academic preparedness and computing participation in CS students' success, and (2) to what extent these factors impact the academic success of URG students.

Our findings reveal significant correlations between underperformance in CS1 and inadequate pre-college math preparation, consistent with existing literature. Moreover, our results show that most URG students also lack prior programming exposure and peer communities within their field (study groups or clubs).

These insights highlight the urgency to consider curriculum and academic preparation modifications to establish a strong math and programming foundation for all students, which can guide them toward greater success in CS.

ACM Reference Format:
Barbara Martinez Neda, Flor Morales, Kitana Carbajal Juarez, Jennifer Wong-Ma, and Sergio Gago-Masague. 2024. Impacts of Academic Preparedness on CS1 Performance. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635614

1 INTRODUCTION/GOALS

Computer Science programs' popularity and selectivity have increased in US institutions [1, 2]. Despite increased CS admissions selectivity, [3] showed continued high academic probation rates in CS at UCI, and even higher rates for URG students. Probation students are informed about available resources and typically receive no direct academic support beyond an academic contract [4].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635614

Previous work offered valuable insights into understanding and modeling students' experiences with academic probation [3]. We found that CS students were most likely to enter probation during their first year and those who entered probation were twice as likely to leave CS. In addition, 57% of students and 78% of URG students experienced probation during their careers. These results highlight the "leaky pipeline" in higher education where URG students succeed at lower rates than their peers [5]. Similarly, [6] found that having a strong math background before entering college was significant to succeed in CS.

Work in [7] revealed that high-achieving high school students attributed their success to minimal effort, leading to a lack of essential university-level skills. [8] found that cramming led to poorer performance in subsequent related courses. Additionally, [9] highlighted that lack of personal direction and clear goals results in unsuitable program choices and mismatched expectations, thus contributing to academic probation [10]. Driven by these findings we designed a comprehensive survey to capture a holistic view of student challenges in a CS1 course in an effort to answer the following research question: *To what extent does academic preparedness contribute to lower student performance in CS1, and do these results differ for URG students?*

In this poster, we will share the results regarding students' academic behaviors. We believe the insights will aid the community in identifying potential curriculum modifications and support the development of inclusive academic support structures.

2 APPROACH

Our comprehensive survey consisted of a total of 92 questions. In this poster, we focus on the responses to 12 general information questions and 24 academic behavior questions. A subset of the survey questions are presented in Tables 1 and 2. This study targeted students enrolled in a CS1 course during the 2022-2023 academic year. The survey was conducted during the first three weeks of the term to ensure that responses better reflected initial experiences with programming and the field. We performed Chi-square tests of independence, cross-tabulation analysis and calculated Cramer's V between survey questions and course performance [11].

3 RESULTS

Table 1 summarizes the academic factors associated with student performance. We found the math placement level (Math Preparation), thoughts of leaving the CS major (Leaving CS), students'

Factor	Survey Question	p-value	Chi-Square	Cramer's V
Leaving CS	(L) Since entering my computing major/minor I have seriously thought about changing to a different major or minor.	p<0.001	14.163	0.214
Math Preparation	(SA) Which of the following courses was or will be your first math course at UCI?	p<0.001	34.302	0.334
Passing Confidence	(L) How confident are you that you will pass the class with a C or a higher grade?	0.0046	10.765	0.187
Programming Experience	(SA) Select your prior programming experiences	0.023	11.323	0.192

Table 1: Factors assoiciated with performance, Chi-square test results, Cramer's V & corresponding survey questions: (L) Likert questions, (SA) Select All that Apply

Factor	Survey Question	p-value	Chi-Square	Cramer's V
Programming Course	Did you have this prior programming experience: high school class or elective.	p<0.001	5.704	0.219
Math Preparation	(SA) Which of the following courses was or will be your first math course at UCI?	p<0.001	50.59	0.405
Passing Satisfaction	(L) I am satisfied with passing all my classes regardless of the letter grade.	p<0.001	14.835	0.219
Club Invovement	I am involved in clubs.	0.022	11.489	0.193
CS Study Groups	(L) I have a study group I can go to for help and support for the course material.	0.013	6.156	0.141

Table 2: Survey questions relevant to URG status, Chi-square test results, Cramer's V & corresponding survey questions: (L) Likert questions

programming experience (Programming Experience), and their confidence that they will pass the class with a C or higher (Passing Confidence) are significantly correlated to performance in CS1.

Consistent with prior work in this area, the factor with the strongest effect on course performance was Math Preparation. This question captured the academic level of the first mathematics course taken at UCI, reflecting students' level of pre-college Math Preparation. We found that students who were required to take pre-calculus at UCI obtained lower grades in CS1 (47% obtained a grade of C+ or below, vs 26% for the average). Considering Leaving CS was the second strongest factor indicating lower performance. Similarly, students who reported less Passing Confidence and less Programming Experience were also strongly linked to lower grades in CS1.

In addition, we explored URG student experiences to address high rates of academic probation among Black and Latino students, revealing that URG-identifying students achieved lower grades compared to their peers. More importantly, URG student status had strong associations with a number of additional factors in academic areas that do not affect other groups, presented in Table 2. URG students reported low participation in high school programming courses, potentially indicating the importance of a CS0 course. Once enrolled at R1U, URG students expressed higher interest in enrolling in the Small Group Tutoring Programs for the CS1 course. Additionally, URG students often reported fewer study groups in their courses. These two behaviors highlight the importance learning communities have on student academic success.

4 CONCLUSION & FUTURE WORK

Our findings reinforced results from prior work indicating that stronger math preparation leads to higher academic success in CS [3, 6]. In addition to the academic factors impacting all students, we found that URG students often face several additional factors that further impact their academic performance. Notably, URG students often have less Club Involvement and CS Study Groups than their peers while also having less math and programming course experiences. This highlights the need to understand the unique experiences URG students face that may have an effect on their performance. For example, we found that URG students are 2.6 times more likely to report a lack of math preparation than their peers.

We plan to continue this study by interviewing students, counselors, and instructors based on our current findings to further understand academic behaviors that impact student performance in CS1. We believe the insights will aid the community in identifying potential curriculum modifications and support the development of inclusive academic support structures.

ACKNOWLEDGEMENTS

This work has been partially supported by the Division of Teaching Excellence and Innovation at UCI through a Provost Faculty Fellowship. We thank Vice Provost Michael Dennin for guidance and assistance in obtaining student datasets. We also thank the School of ICS for providing student support that facilitated this work.

REFERENCES

[1] U.S. Department of Education. Institute of Education Sciences, National Center for Education Statistics. *Digest of Education Statistics, 2022*. 2022. URL: https://nces.ed.gov/programs/digest/d22/tables/dt22_322.10.asp.

[2] *Undergraduate Admissions Dashboard*. 2022. URL: https://datahub.oapir.uci.edu/Undergraduate-Admissions-Dashboard.php.

[3] Barbara Martinez Neda et al. "Revisiting Academic Probation in CS: At-Risk Indicators and Impact on Student Success". In: *J. Comput. Sci. Coll.* 38.10 (June 2023), pp. 8–16. ISSN: 1937-4771.

[4] UC Irvine. Donald Bren School of Information and Computer Sciences. *Undergraduate Student Policies*. URL: https://ics.uci.edu/academics/undergraduate-programs/majors-minors/undergraduate-student-policies/.

[5] Mica Estrada et al. "Improving underrepresented minority student persistence in stem". In: *CBE Life Sciences Education* 15.3 (Sept. 2016). ISSN: 19317913. DOI: 10.1187/cbe.16-01-0038. URL: https://www.lifescied.org/doi/10.1187/cbe.16-01-0038.

[6] Chen Chen et al. "High School Calculus and Computer Science Course Taking as Predictors of Success in Introductory College Computer Science". In: 21.1 (2021). URL: https://doi.org/10.1145/3433169.

[7] Megan Balduf. "Underachievement Among College Students". In: *Journal of Advanced Academics* 20.2 (2009), pp. 274–294. URL: https://doi.org/10.1177/1932202X0902000204.

[8] Shelby H. McIntyre and J. Michael Munson. "Exploring Cramming: Student Behaviors, Beliefs, and Learning Retention in the Principles of Marketing Course". In: *Journal of Marketing Education* 30.3 (2008), pp. 226–243. URL: https://doi.org/10.1177/0273475308321819.

[9] James Cindy L. and Graham Sarah. "An Empirical Study of Students on Academic Probation". In: *Journal of the First-Year Experience and Students in Transition* 22.2 (2010). URL: https://eric.ed.gov/?id=EJ906631.

[10] Chris Maharaj et al. "Help Me Else I Might Fail! Solutions for Academically Challenged Engineering Students". In: *Journal of College Student Retention: Research, Theory & Practice* 23.3 (2021), pp. 607–635. URL: https://doi.org/10.1177/1521025119865747.

[11] Theresa Marchant-Shapiro. *Statistics for Political Analysis: Understanding the Numbers*. 2015. DOI: 10.4135/9781483395418NV-0. URL: https://methods.sagepub.com/book/statistics-for-political-analysis-understanding-the-numbers.

More Than a Checklist: Redesigning a UI/UX Curriculum to Emphasize Accessibility

Talie Massachi
talie_massachi@brown.edu
Brown University
Providence, RI, USA

ABSTRACT

Despite widespread recognition of its value and importance, accessibility tends to be interpreted as a niche topic by students in many User Interfaces and User Experience (UI/UX) courses. In response, we advocate for explicitly positioning accessibility as a fundamental building block in UI/UX education. We present one implementation of this via early inclusion of accessibility and inclusivity principles, supplemented by strategic reinforcement across course modules. Early student reception of this intervention has been positive.

ACM Reference Format:
Talie Massachi. 2024. More Than a Checklist: Redesigning a UI/UX Curriculum to Emphasize Accessibility. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635534

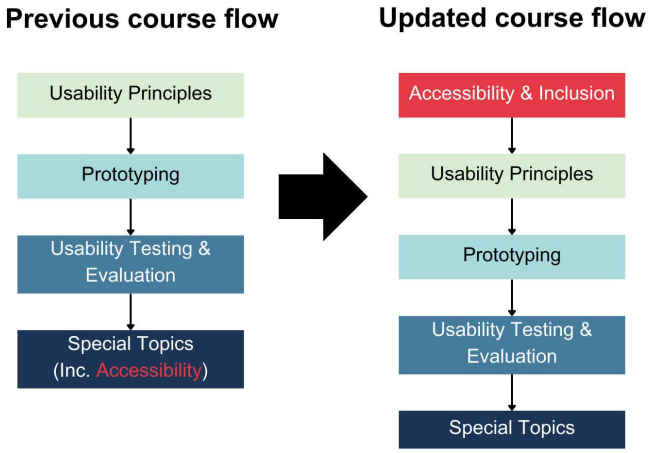


Figure 1: The change in course flow from before (left) to after (right) our changes. Our changes included explicitly highlighting accessibility and inclusion as core values of the course, and in tandem moving discussions on these topics to the beginning of the course.

1 INTRODUCTION

Despite near unanimous agreement about the value and importance of accessible software [3, 4, 8, 10], User Interfaces and User Experience (UI/UX) curricula tend to treat accessibility and inclusivity as an afterthought. Indeed, publicly available online syllabi from several major universities, including our own, wait to explicitly address inclusive design until the very end of the course, if at all, (e.g., [2, 5–7, 9]). By placing these discussions at the end of the schedule, we are implicitly training budding UX professionals that accessibility and inclusivity principles are inherently separate from the fundamentals of design.

2 STARTING WITH CORE VALUES

UI/UX courses at our university aim to introduce students to the principles of user experience design, including methods for designing and prototyping interfaces, and user interface evaluation. These UI/UX curricula have historically started with introductory topics such as affordances, mental models, and prototyping. In doing so we implicitly introduce these concepts as "building blocks" upon which UI/UX and design principles are built. The core values of inclusivity and accessibility are implicitly intertwined with these other building blocks. In previous versions of this course, these values are promoted through required compliance with WCAG AA accessibility guidelines [1] in several assignments throughout the semester.

However, explicit discussions about accessibility and inclusive design do not take place until the very end of the course. This reinforces student conceptions of these topics as "niche", and incidental to UI/UX as a field, as evidenced by discussions with students. By the end of the course, we note that students view these guidelines as constraints upon their designs rather than fundamental principles of design.

To address this gap in understandings, we made the core values of accessibility and inclusive design explicit. We took this further by positioning these topics in front of other building blocks as the first addressed in the course through both lecture and in-class discussion. In doing so, future student approaches to design were colored by exposure to this new central building block.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635534

3 CONTEXTUALIZING ACCESSIBILITY AND INCLUSIVITY

The introduction of accessibility and inclusivity early in our redesigned course allowed us to contextualize these values in each of the following modules. This reiteration followed established strategies we have historically found useful when reinforcing other key design building blocks.

For example, the first assignment of the *original* course asked students to perform a contextual inquiry of a selected interface and then build a set of personas based on the gathered information. In doing so, students were reminded to consider usability principles (e.g., efficiency, mental vs. conceptual models) while observing and analyzing interactions. The redesigned assignment reminded students to consider diverse participants while gathering information, including participants with accessibility needs. This was strengthened by the inclusion of exercises designed to encourage critical examination of student-created personas [1].

4 STUDENT RESPONSE

In bringing accessibility and inclusive design forward as explicit core values, observation of in-class activities showed students engaging actively with these topics alongside other usability principles. Informal interviews during the semester and mid-semester and end-of-semester student feedback showed that students actively strove to understand how designing for accessibility impacted different steps of the design process.

During critiques, students proactively brought up and addressed accessibility concerns without prompting. Students still noted issues with other principles such as clarity of visual metaphor, memorability, and efficiency. However, students also pointed out potential accessibility improvements, referencing concepts such as visual contrast and font readability without prompting from instructors. In final course projects, students also readily discussed accessibility and inclusion-based design choices with little prompting.

Overall, students seemed to value the centering of accessibility as a building block. Student end-of-semester feedback included appreciation for the "focus on accessibility and inclusive design". With some students mentioning a desire for "more on accessibility. [I] still am trying to learn how to make these considerations an integral part of my design process".

ACKNOWLEDGMENTS

A sincere thank you to Siddhartha Prasad and Elijah Rivera for their feedback and assistance in preparing this work.

REFERENCES

[1] World Wide Web Consortium. 2023. *WCAG 2 Overview.* https://www.w3.org/WAI/standards-guidelines/wcag/
[2] MIT Open Courseware. 2011. User Interface Design and Implementation. https://ocw.mit.edu/courses/6-831-user-interface-design-and-implementation-spring-2011/pages/calendar/. Accessed January 26, 2024.
[3] Yavuz Inal, Frode Guribye, Dorina Rajanen, Mikko Rajanen, and Mattias Rost. 2020. Perspectives and practices of digital accessibility: A survey of user experience professionals in nordic countries. In *Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society*. 1–11.
[4] Steven I. Jacobs. 1999. *Fueling the Creation of New Electronic Curbcuts.* http://www.accessiblesociety.org/topics/technology/eleccurbcut.htm
[5] USC Viterbi School of Engineering. 2018. ITP 310 Design for User Experience. https://web-app.usc.edu/soc/syllabus/20181/32020.pdf. Accessed January 26, 2024.
[6] University of Texis at Austin McCombs School of Business. 2024. Post Graduate Program in User Experience Design: UI/UX for Data-Driven Business Applications. https://onlineexeced.mccombs.utexas.edu/pg-program-online-uiux-design-course. Accessed January 26, 2024.
[7] University of Texis at Austin School of Information. 2021. I310U Introduction to User Experience Design. https://www.ischool.utexas.edu/sites/default/files/webform/syllabi/Intro_UXD_I310U_Syllabus_Spring2021_pub.pdf. Accessed January 26, 2024.
[8] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O Wobbrock. 2020. An epidemiology-inspired large-scale analysis of android app accessibility. *ACM Transactions on Accessible Computing (TACCESS)* 13, 1 (2020), 1–36.
[9] Brown University Computer Science. 2022. CSCI 1300 User Interfaces and User Experience. https://cs.brown.edu/courses/csci1300/fall2020/. Accessed January 26, 2024.
[10] John M Slatin and Sharron Rush. 2003. *Maximum accessibility: Making your web site more usable for everyone.* Addison-Wesley Professional.

1. Example questions include *Are there any aspects of your personas that may represent inherent biases?* and *What parts of the persona-building process could change to alleviate some of these biases?*

Developing Computational Thinking in Middle School Music Technology Classrooms

Lauren McCall
Center for Music Technology
Georgia Institute for Technology
Atlanta, Georgia
lmccall7@gatech.edu

Brittney Allen
Center for Music Technology
Georgia Institute of Technology
Atlanta, Georgia
brittneyjuliet@gatech.edu

Jason Freeman
Center for Music Technology
Georgia Institute of Technology
Atlanta, Georgia
jason.freeman@gatech.edu

Stephen Garrett
Center for Music Technology
Georgia Institute of Technology
Atlanta, Georgia
garrett@gatech.edu

Sabrina Grossman
Center for Education Integrating Science, Math, and Computing
Georgia Institute of Technology
Atlanta, Georgia
sabrina.grossman@ceismc.gatech.edu

Jed Paz
Center for Education Integrating Science, Math, and Computing
Georgia Institute of Technology
Atlanta, Georgia
jpaz6@gatech.edu

Doug Edwards
Center for Education Integrating Science, Math and Computing
Georgia Institute of Technology
Atlanta, Georgia
doug.edwards@ceismc.gatech.edu

Tom McKlin
The Findings Group
Decatur, Georgia
tom@thefindingsgroup.org

Taneisha Lee
The Findings Group
Decatur, Georgia
taneisha@thefindingsgroup.org

ABSTRACT

To engage diverse populations of students who may not self-select into computing courses, a curriculum for a middle school music technology + computer science course that addresses learning standards for both subjects was developed and deployed. Students who engage with the curriculum learn modern music production techniques and computational thinking concepts. This is through a mix of traditional approaches to music technology education (digital audio workstations) and computational approaches via a culturally relevant learning platform that introduces students to coding through music production and remixing. This poster reflects on the last two years of curriculum design and deployment, teacher training, and student and educator engagement and feedback to provide insight into the teaching (and learning) of computational thinking in the music technology classroom.

CCS CONCEPTS

• Applied computing → Sound and music computing; • Human-centered computing → Participatory design; • Social and professional topics → K-12 education; Computational thinking.

KEYWORDS

music, computing, middle school, interest formation, persistence, broadening participation

ACM Reference format:
Lauren McCall, Brittney Allen, Jason Freeman, Stephen Garrett, Sabrina Grossman, Jed Paz, Doug Edwards, Tom McKlin, and Taneisha Lee. 2024. Developing Computational Thinking in Middle School Music Technology Classrooms. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635527

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

© 2024 Copyright is held by the owner/author(s).

ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635527

1 INTRODUCTION

Although computational thinking has become an essential skill in today's technologically advanced workforce, the engagement and retention of underrepresented groups in computer science is lagging. In the latest Gallup/Google report on Computer Science Education, only 31% of girls agree that computer science is important to learn [3]. While interest in computer science has increased among underrepresented minorities, access to dedicated courses lags that of white students [3]. As recently as 2016, only 1% of high schoolers in Georgia completed a course including computational thinking skills. To address this, in 2019, Georgia passed legislation requiring all middle and high school schools to offer computer science courses by 2024. In this poster, we describe a researcher-practitioner partnership between a research university and a K-12 school district to create, deploy, and study the potential of integrating computer science education into the curriculum of a middle school music technology course. The integration of computational thinking with music technology

classes is an educational method used in several computing education platforms to increase engagement in computational thinking [1, 2, 4]. By using a computer science + music framework, it is our hope to broaden participation by expanding availability to CS and engaging students in CS education who may not self-select into more traditional computing courses. In the following sections, we outline the curriculum framework, describe the methodology and results of an initial research study.

2 CURRICULUM

The curriculum for this project leverages both traditional music production software, in which we linked critical elements of music technology to computational thinking principles, and a widely used computational music platform, EarSketch, previously developed by some of the authors. With EarSketch, students write code to programmatically generate music using an interface design similar to traditional music production software (Figure 1). The curriculum combines existing materials from the Georgia State curriculum for middle school beginning music technology courses with materials created by a team of music technology and computing education researchers and practitioners specifically for this project. One example of a lesson developed for this curriculum involved students creating musical rhythms with code. With this lesson, CS concepts, including functions and strings, were used. The lesson involved a progression from developing rhythms in a DAW to mapping these rhythms into strings through a rhythm-generating function in EarSketch.

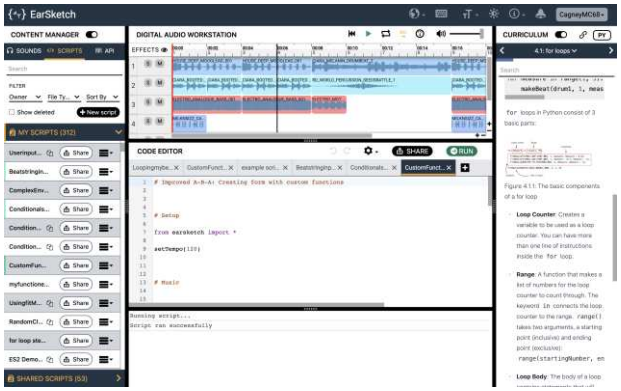


Figure 1. EarSketch Interface

3 METHODOLOGY

During the 2022-2023 academic school year, we conducted a preliminary study in two middle schools, enlisting the help of two music technology teachers, each acting as a practitioner partner at their respective school. Earlier, in the spring and summer of 2022, the teachers participated in professional development to learn EarSketch [5], the music+CS learning platform used in parts of the curriculum. They worked directly with the research team to collaboratively create the curriculum described in the previous section for their beginning music technology courses. Students in all sections of the beginning music technology class at both schools were invited to participate in a research study that included a pre- and post-survey and four focus groups with a subset of consenting students.

4 RESULTS

Our analysis suggests that a) students not already interested in music technology or programming do not connect with the material as well as students with prior interest; b) music technology educators require targeted training and support in pedagogical content knowledge in computing; and c) greater intentional connection between music and computer science throughout the curriculum has the potential to spur further student engagement. Following this study, we updated the curriculum by adding more diverse representations of professionals in the music and computing industries and developing a lesson on music in the digital age that connects to a wide range of student interests. We updated the professional development materials by providing revised educational materials with extensive scaffolding. Additionally, we offered teacher support through an online community and morning professional development sessions during the school year. We delivered the main aspects of the professional development to a new cohort of five middle school music technology teachers in the summer of 2023, and they – along with the original two teachers – all began implementing the new curriculum in their classrooms in August 2023. We are currently conducting an expanded study with these seven schools in both the fall 2023 and spring 2024 semesters, in which we are collecting quantitative and qualitative data similar to the prior year.

REFERENCES

[1] Andrew R. Brown and S. Alex Ruthmann. 2020. Scratch Music Projects. Oxford University Press.
[2] Bill Manaris, Blake Stevens, and Andrew R. Brown. 2016. JythonMusic: An environment for teaching algorithmic music composition, dynamic coding and musical performativity. Journal of Music, Technology & Education 9, 1 (May 2016), 33–56. https://doi.org/10.1386/jmte.9.1.33_1 Publisher: Intellect.
[3] Google LLC and Gallup Inc. 2020. Current Perspectives and Continuing Challenges in Computer Science Education in US K-12 Schools.
[4] William Payne and Alex Ruthmann. 2019. Music making in Scratch: High floors, low ceilings, and narrow walls. Journal of Interactive Technology and Pedagogy no. 15 (2019).
[5] John Peterson and Greg Haynes. 2017. Integrating Computer Science into Music Education. In Proceedings of the Education (Seattle, Washington, USA) (SIGCSE '17). Association for Computing Machinery, New York, NY, USA, 459–464. https://doi.org/10.1145/3017680.3017767

Unintentional Barriers for AP Computer Science Principles: A Course Designed for Every Student

Emma L. McDaniel
Georgia State University
Atlanta, GA, USA
emcdaniel10@gsu.edu

Aaja Christie
Georgia State University
Atlanta, GA, USA
achristie3@gsu.edu

Anu G. Bourgeois
Georgia State University
Atlanta, GA, USA
abourgeois@gsu.edu

ABSTRACT

In 2016, College Board launched Advanced Placement (AP) Computer Science Principles (CSP) as a college-level introductory course to the breadth of Computer Science (CS). This was a joint effort with National Science Foundation to produce a course with a lower barrier of entry to create new pathways of introducing CS to traditionally underrepresented groups in computing; a high school course designed for every student. The sole recommended prerequisite for the course is Algebra I; however, we found that often additional prerequisites and barriers have been imposed. These additional requirements can inadvertently undermine the inclusive vision that underpinned the course's creation. We focus on public high schools in Georgia and discuss different policy barriers that can impact enrollment in AP CSP.

KEYWORDS

Broadening Participation in Computing, Computer Science Education, K-12 education, AP Computer Science

ACM Reference Format:
Emma L. McDaniel, Aaja Christie, and Anu G. Bourgeois. 2024. Unintentional Barriers for AP Computer Science Principles: A Course Designed for Every Student. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635499

1 INTRODUCTION

The intention of College Board's Advanced Placement (AP) Computer Science Principles (CSP) is to be a gateway course that introduces students to the breadth of Computer Science (CS) [4, 9]. Their AP Equity and Access Policy states "We encourage the elimination of barriers that restrict access to AP for students from ethnic, racial, and socioeconomic groups that have been traditionally underrepresented. Schools should make every effort to ensure that their AP classes reflect the diversity of their student population" [5].

AP Computer Science A (CSA) is another, preexisting, AP computing course. Since the introduction of AP CSP, there has been a nationally significant increase of students enrolling in AP CS courses. Many of the newer enrollment belong to traditionally underrepresented groups (URGs) in computing, which are students belonging to the race/ethnicity groups of Black, Hispanic, Native American, and Two or More Races, as well as women [12, 14].

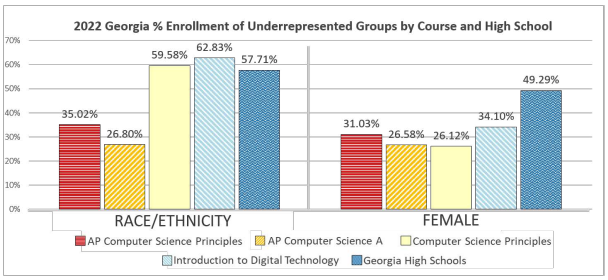


Figure 1: GA Enrollment data of students belonging to URGs

Havard and Howard [10] state that the "depth and rigor" in AP CSP is less that that of AP CSA, and focuses on the breadth of CS, whilst AP CSA focuses on the technical aspects of programming and algorithms. Further, AP CSP does not require the use of a traditional text-based coding language, and the specified deliverables for any code are minimal: a purpose, an input, and an output [4]. AP CSP is undoubtedly an introductory course to CS. So why is it that for many schools, there are prerequisites or policy barriers that limit access and/or availability of AP CSP?

We focus on the state of Georgia (GA) for this paper. However, through random inspection of course requirements across a sampling of states, we found that these barriers are not unique to schools in GA. We first look at the enrollment of students belonging to URGs in four computing courses and across GA high schools. Fig. 1 [1] shows the enrollments in AP CSP, AP CSA, CSP, and Introduction to Digital Technology (IDT), as well as enrollment data of students belonging to URGs for all GA public high schools in 2022. CSP is the non-AP iteration of AP CSP, which often covers the same material as AP CSP, but lacks the option for students to take the AP exam to earn college credit. IDT[2] [7] is a course that covers formal business communication and digital literacy.

Clearly, the enrollment in both AP CS classes do not reflect the racial/ethnic diversity and gender of the enrollment across GA high schools. There is also a significant difference between students enrolled in IDT and CSP (not AP) to that of AP CSP and AP CSA. The high enrollment in the non-AP courses could indicate that there is a high interest in computing among students belonging to race/ethnicity URGs, which is corroborated by the 2020 Google Gallup Poll [8]. The following section examines specific barriers to AP CSP that are frequently imposed in GA. As a result, these

[1]Note Fig. 1 utilizes the term 'Female', which is an artifact of the data collection and the authors understand this does not always accurately represent gender identity.
[2]As of 2023, the IDT course is being renamed to Introduction to Software Technology.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635499

barriers can unintentionally limit the accessibility of AP CSP and thus, exposure to basic computer science concepts.

2 BARRIERS

We concentrate on four policy barriers designated and implemented at the school, district, and state levels across Georgia (GA). The methodology for selecting these particular barriers are from the authors' experience of collaboration with CS4GA (a collective impact effort to bring quality CS education to all K-12 children in GA), training high school CS teachers, and previous research in the domain [2].

Multiple Levels of Approval for Advanced Placement: In GA high schools, requirements to enroll into AP classes usually include parental permission on an AP Rigor Form and/or a teacher's recommendation. These are *reasonable* requirements for typical AP classes, but not necessarily relevant for AP CSP. The form describes how material is presented at a faster pace and at a college level, with expectations of heavier workloads for the class. Consequently, this language may discourage many students from taking AP CSP. Additionally, a parent may be reluctant to give approval if they are unaware of the actual structure of AP CSP and its introductory nature. Regarding teacher permission, biases have been shown in providing recommendations unequally for white students [11].

Additional Prerequisite Courses: The AP CSP course description recommends successful completion of Algebra I as a prerequisite [4]. However, across the five largest school districts in GA, there are often more prerequisites, with the most common being the IDT/IST course. Recently, some districts have changed their policy on a prerequisite of IDT, so only Algebra I and/or being in 10-12th grade is needed, which is in line with College Board recommendations. AP CSP (or non-AP CSP) counts as an elective and if students must take a prerequisite, then to take CSP, it will require two credits to explore this introduction to computing. Enrollment in AP CSP is significantly lower than in IDT (Fig. 1), implying that IDT is a bottleneck for enrolling in AP CSP. Extra prerequisites are unnecessary for an introductory course like AP CSP.

Career and Technical Education Pathways: The Strengthening Career and Technical Education (CTE) for the 21st Century Act was signed as federal law in 2018 [13]. For many states, CTE programs are implemented by designating elective courses to belong to a specific career cluster or pathway. In GA, the AP CSP course is only included in the Information Technology cluster (out of 17 clusters), and is the second in a sequence of three courses, with IDT being the first [7]. Constricting AP CSP to a path of technology related careers limits exposure to computing, as it places it in a pipeline that may not appeal to many students. Also, to take AP CSP, students must commit two of their limited electives to a pathway about which they might have uncertainties. AP CSP is a course designed for exploration, but it is instead restricted to career pathways. A small number of clusters have a specific technology course included, however, AP CSP is a general course applicable to any domain.

CS Teacher Certification: Another factor that can hinder access to AP CSP is the lack of teachers. According to AP Central, teachers do not need to be CS certified or have prior experience [3]. The College Board only has an expectation that the teacher has basic knowledge and can be from any background, with the only requirement being the teacher undergoes a course audit to confirm necessary topics and objectives of the course are covered. Further restrictions on who can teach CS courses are typically applied by individual states. Zarch et al. [15] detail policy decisions that have unintended consequences in broadening participation in computing; one example is the Georgia Professional Commission requiring CS courses (including AP CSP) be taught by a CS certified teacher. To gain certification in GA, teachers must either pass the Georgia Assessments for Certification for Educators (GACE) CS exam or complete a CS endorsement program. This potentially imposes notable constraints on the availability of AP CSP, particularly affecting schools in rural and lower-income areas disproportionately [1, 6].

3 IMPLICATIONS FOR THE CS COMMUNITY

The Advanced Placement (AP) Computer Science Principles (CSP) course provides a means for access to exploring CS. We found barriers that unintentionally restrict access to the course. To overcome the stigma of AP courses, some schools offer a non-AP CSP course to remove the stigma of an AP course, but this prohibits gaining college credit. Ideally, AP CSP should be a general elective, clear of: AP enrollment barriers, CS certification for instructors, and prerequisites. Additionally, if career pathways continue to be emphasized, then AP CSP should be incorporated into diverse tracks beyond the technology career pathway.

REFERENCES

[1] Broneak, C., and Rosato, J. Experiences of rural CS principles educators. In *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)* (2021), IEEE, pp. 1–2.

[2] Christie, A., McDaniel, E. L., Tiwari, C., Mikler, A. R., and Bourgeois, A. G. How to Allocate and Assess Equity of CS Education Resources? In *Equity and Access in Algorithms, Mechanisms, and Optimization* (2022).

[3] College Board. Do I need specific experience to teach AP Computer Science Principles? [Online] Date Accessed: August 9, 2023.

[4] College Board. AP Computer Science Principles: Course and Exam Description, 2020. [Online] Date Accessed: February 23, 2023.

[5] College Board. Broadening Access to Advanced Placement: A Toolkit for Educators and School Leaders Created by AP Teachers, 2022.

[6] Eckert, S. A. What do teaching qualifications mean in urban schools? A mixed-methods study of teacher preparation and qualification. *Journal of Teacher Education 64*, 1 (2013), 75–89.

[7] Georgia Department of Education. Information Technology Cluster. https://www.gadoe.org/Curriculum-Instruction-and-Assessment/CTAE/Pages/cluster-IT.aspx, 2023.

[8] Google, & Gallup. Current perspectives and continuing challenges in computer science education in US K-12 schools. https://csedu.gallup.com/home.aspx, 2020. [Online] Accessed February 23, 2023.

[9] Guzdial, M. Drumming up support for AP CS principles. *Communications of the ACM 59*, 2 (2016), 12–13.

[10] Havard, D. D., and Howard, K. E. All Advanced Placement (AP) Computer Science is Not Created Equal: A comparison of AP Computer Science A and Computer Science Principles. *Journal of Computer Science Integration 2*, 1 (2019).

[11] Lewis, A. E., and Diamond, J. B. *Despite the best intentions: How racial inequality thrives in good schools.* Oxford University Press, 2015.

[12] Mark, J., and Klein, K. Beauty and Joy of Computing: 2016-17 Findings from an AP CS Principles course. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (2019), pp. 627–633.

[13] Perkins Collaborative Resource Network. PCRN: Perkins V. https://cte.ed.gov/legislation/perkins-v, 2018.

[14] Sax, L. J., Newhouse, K. N., Goode, J., Skorodinsky, M., Nakajima, T. M., and Sendowski, M. Does AP CS Principles Broaden Participation in Computing?: An Analysis of APCSA and APCSP Participants. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (2020), pp. 542–548.

[15] Zarch, R., Dunton, S., Childs, J., and Leftwich, A. Through the Looking Glass: Computer Science Education and the Unintended Consequences of Broadening Participation Policy Efforts. In *2020 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)* (2020), vol. 1, IEEE, pp. 1–4.

Validation of a Bebras-Based Test to Assess Computational Thinking Abilities in First-Year College Students

Federico Meza
federico.meza@usm.cl
Univ. Técnica Federico Santa María
Santiago, Chile

Andrea Vásquez
andrea.vasquezg@usm.cl
Univ. Técnica Federico Santa María
Valparaíso, Chile

Daniel San Martín
daniel.sanmartinr@usm.cl
Univ. Técnica Federico Santa María
Valparaíso, Chile

ABSTRACT

The lack of clarity about the definition of Computational Thinking (CT) undermines its assessment and the formulation of effective learning strategies for its development. We propose an assessment tool in Spanish, designed by carefully selecting Bebras tasks, to measure four specific CT skills in first-year university students with no prior programming experience. The test was validated with a sample of 980 students from a Latin American university. We found a high, positive correlation between test results and Math test results from the National University Selection System, and a medium, positive correlation between test results and grades from the midterm exam in a Programming course. An acceptable level of internal consistency was found (Cronbach's $\alpha = 0.70$). We examined the validity of the test using Classical Test Theory. One question showed poor discrimination potential. Its elimination from the test increased the internal consistency, so we propose to replace it.

KEYWORDS

assessment, validation, computational thinking, bebras

ACM Reference Format:
Federico Meza, Andrea Vásquez, and Daniel San Martín. 2024. Validation of a Bebras-Based Test to Assess Computational Thinking Abilities in First-Year College Students. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635523

1 BACKGROUND AND RELATED WORK

Computational Thinking (CT) comprises essential problem-solving skills. However, due to the absence of a consensus on the specific components of CT, its evaluation process remains ambiguous [13]. This lack of clarity hampers the formulation of effective learning strategies to facilitate CT development.

Numerous assessment instruments for CT exist, each following distinct approaches and operational definitions [12, 13]. Most of these tools either revolve around programming tasks or primarily foster algorithmic thinking. Furthermore, these assessments are typically designed to cater to a wide age range, predominantly targeting school-aged children [13].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635523

The Bebras Challenge, originally created to promote students' interest in computer science careers [5], possesses attributes that render it a suitable CT evaluation tool [11]. Bebras Challenges have been used in previous experiences as a tool for CT assessment, customizing them to meet specific objectives and yielding diverse outcomes [3, 6, 7, 9].

2 APPROACH AND UNIQUENESS

We propose an assessment tool designed to measure four Computational Thinking (CT) skills: abstraction, generalization, decomposition, and algorithmic thinking. Our target audience is incoming university students with no prior programming experience. This instrument was designed for flexible use, allowing both computer-based and non-computer-based administration, with the aim of completing it within an hour, making it suitable for inclusion in regular scheduled classes.

Our proposed instrument was developed by carefully selecting individual questions or "tasks" from various Bebras Challenges. This approach ensured a balanced evaluation of each CT skill, maintaining internal consistency, appropriate levels of difficulty, and discrimination power [8]. Unlike prior studies [2, 4, 14], we did not use complete Bebras Challenges but instead curated specific tasks to assess each skill across different levels of complexity.

Our research team meticulously reviewed over 500 tasks, spanning different age groups. From this extensive review, we selected 57 tasks based on two criteria: their alignment with the aforementioned CT skills and their perceived difficulty for our target student population. Two researchers independently classified each task based on the CT skills they evaluated, identifying the most prominent skill for each. Additionally, they rated task difficulty.

Drawing from our experience as instructors of CS1 courses, we estimated that first-year university students could respond to approximately 12 questions within an hour. Using this criterion, we further refined our selection from the initial 57 tasks to include three different tasks of varying difficulty levels for each skill.

The 12 selected questions were translated into Spanish. To ensure linguistic clarity and appropriateness, we conducted a think-aloud session with three senior high school students who had attended a summer Programming bootcamp.

To validate our instrument, we administered it to 1,211 first-year Engineering students enrolled in a Programming course at the Universidad Técnica Federico Santa María in Chile. We examined correlations between the test scores, the National University Selection Test scores, course midterm grade, and course final grade.

To evaluate internal consistency, we applied Cronbach's Alpha. Validity was assessed using difficulty and discrimination indices from Classical Test Theory [1].

3 RESULTS AND CONTRIBUTION

Mean	Median	Mode	SD	Min	Max
55.37	58.33	66.67	22.82	0	100

Table 1: Description of the test results

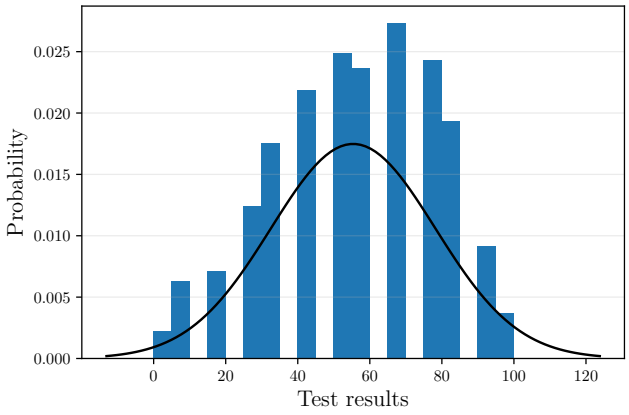


Figure 1: Distribution of the test results

Out of the 1,211 individuals who took the test, 231 were excluded because they were retaking the course. Thus, the number of responses considered in the analysis was 980. Table 1 displays the descriptive statistics of the test results. The distribution of the results can be observed in the histogram of Figure 1.

We found a high, positive correlation ($r = 0.61$, $p < 0.001$) between test results and Math test results from the National University Selection System. This is consistent with previous results suggesting that students with stronger mathematical abilities often do better in CT tests [10]. On the other hand, test results and grades from the midterm exam show a medium, positive correlation ($r = 0.5$, $p < 0.001$). Finally, test results and final grades also exhibit a medium, positive correlation ($r = 0.45$, $p < .001$).

To assess validity we computed the difficulty and discrimination indices from Classical Test Theory. Results are summarized in Table 2. Values that suggest that attention should be paid to the items are highlighted in color. For the difficulty index, question Q4 obtained a very low value, which could indicate that the question is too difficult for the target population. On the other hand, Q7 obtained a very high index, suggesting that it might too easy. Regarding the discrimination index, Q1 obtained an index that raises concerns about its potential discrimination. The rest of the questions obtained values that are considered good.

To evaluate internal consistency we computed Cronbach's Alpha, obtaining a value of $\alpha = 0.70$, which is considered acceptable. Since three questions yielded indices that require attention, a sensitivity analysis was conducted with respect to Cronbach's Alpha by independently removing each of these questions. Only the removal of question Q1 resulted in an increase in Cronbach's Alpha ($\alpha = 0.72$).

In the future, we intend to replace item Q1 with another item that assesses the same CT skill and then repeat the validation process.

Item	Difficulty	Discrimination
Q1	0.37	0.25
Q2	0.59	0.67
Q3	0.56	0.48
Q4	0.31	0.56
Q5	0.61	0.63
Q6	0.38	0.47
Q7	0.81	0.42
Q8	0.68	0.58
Q9	0.49	0.70
Q10	0.67	0.63
Q11	0.68	0.67
Q12	0.50	0.68

Table 2: Results of the Validation Indices

REFERENCES

[1] Mary J. Allen and Wendy M. Yen. 2002. *Introduction to Measurement Theory.* Waveland Press, Long Grove, IL.
[2] Ana Liz Souto O. Araujo, Jucelio S. Santos, Wilkerson L. Andrade, Dalton D. Serey Guerrero, and Valentina Dagienė. 2017. Exploring computational thinking assessment in introductory programming courses. In *2017 IEEE Frontiers in Education Conference (FIE).* 1–9. https://doi.org/10.1109/FIE.2017.8190652
[3] Francisco Bavera, Teresa Quintero, Marcela Daniele, and Flavia Buffarini. 2020. Computational Thinking Skills in Primary Teachers: Evaluation Using Bebras. In *Computer Science – CACIC 2019*, Patricia Pesado and Marcelo Arroyo (Eds.). Springer International Publishing, Cham, 405–415.
[4] Giuseppe Chiazzese, Marco Arrigo, Antonella Chifari, Violetta Lonati, and Crispino Tosto. 2019. Educational Robotics in Primary School: Measuring the Development of Computational Thinking Skills with the Bebras Tasks. *Informatics* 6, 4 (2019). https://doi.org/10.3390/informatics6040043
[5] Valentina Dagienė and Gabrielė Stupurienė. 2016. Bebras - a Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in Education* 15, 1 (2016), 25–44.
[6] Javier del Olmo-Muñoz, Ramón Cózar-Gutiérrez, and José Antonio González-Calero. 2020. Computational thinking through unplugged activities in early years of Primary Education. *Computers & Education* 150 (2020), 103832.
[7] Havva Delal and Diler Oner. 2020. Developing Middle School Students' Computational Thinking Skills Using Unplugged Computing Activities. *Informatics in Education* 19, 1 (2020), 1–13. https://doi.org/10.15388/infedu.2020.01
[8] Filiz Kalelioğlu, Dilek Doğan, and Yasemin Gülbahar. 2022. Snapshot of Computational Thinking in Turkey: A Critique of 2019 Bebras Challenge. *Informatics in Education* 21, 3 (2022), 501–522. https://doi.org/10.15388/infedu.2022.19
[9] James Lockwood and Aidan Mooney. 2018. Developing a Computational Thinking Test Using Bebras Problems. In *CEUR Workshop.*
[10] Aidan Mooney and James Lockwood. 2020. The Analysis of a Novel Computational Thinking Test in a First Year Undergraduate Computer Science Course. *The All Ireland Journal of Teaching and Learning in Higher Education* 12, 1 (2020), 1–27.
[11] Ana Liz Souto Oliveira, Wilkerson L. Andrade, Dalton D. Serey Guerrero, and Monilly Ramos Araujo Melo. 2021. How do Bebras Tasks Explore Algorithmic Thinking Skill in a Computational Thinking Contest?. In *2021 IEEE Frontiers in Education Conference (FIE).* 1–7. https://doi.org/10.1109/FIE49875.2021.9637151
[12] Marcos Román-González, Jesús Moreno-León, and Gregorio Robles. 2019. *Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions.* Springer Singapore, Singapore, 79–98. https://doi.org/10.1007/978-981-13-6528-7_6
[13] Xiaodan Tang, Yue Yin, Qiao Lin, Roxana Hadad, and Xiaoming Zhai. 2020. Assessing computational thinking: A systematic review of empirical studies. *Computers & Education* 148 (2020), 103798.
[14] E M Zamzami, J T Tarigan, N Zendrato, A Muis, A P Yoga, and M Faisal. 2020. Exercising the Students Computational Thinking Ability using Bebras Challenge. *Journal of Physics: Conference Series* 1566, 1 (jun 2020), 012113.

A ONLINE RESOURCES

Anonymized data for the purpose of reproducibility is available in the following repository: https://github.com/dsanmartin/OEA2202. Test questions can be obtained upon request.

Elevating Learning Experiences: Leveraging Large Language Models as Student-Facing Assistants in Discussion Forums

Chancharik Mitra*
University of California, Berkeley
Berkeley, CA, USA
cmitra@berkeley.edu

Mihran Miroyan*
University of California, Berkeley
Berkeley, CA, USA
miroyan.mihran@berkeley.edu

Rishi Jain*
University of California, Berkeley
Berkeley, CA, USA
rishiraij@berkeley.edu

Vedant Kumud
University of California, Berkeley
Berkeley, CA, USA
vkumud@berkeley.edu

Gireeja Ranade
University of California, Berkeley
Berkeley, CA, USA
ranade@eecs.berkeley.edu

Narges Norouzi
University of California, Berkeley
Berkeley, CA, USA
norouzi@berkeley.edu

ABSTRACT

Recent advancements in instruction-tuned large language models offer new potential for enhancing students' experiences in large-scale classes. Deploying LLMs as student-facing assistants, however, presents challenges. Key issues include integrating class-specific content into responses and applying effective pedagogical techniques. This study addresses these challenges through retrieval and prompting techniques, focusing on mitigating hallucinations in LLM-generated responses, a crucial concern in education. Furthermore, practical deployment brings further challenges related to student data privacy and computational constraints. This research strives to enhance the quality and relevance of LLM responses while addressing practical deployment issues, with an emphasis on creating a versatile system for diverse domains and teaching styles.

KEYWORDS

Natural language processing, Educational tools, Discussion forum

ACM Reference Format:
Chancharik Mitra, Mihran Miroyan, Rishi Jain, Vedant Kumud, Gireeja Ranade, and Narges Norouzi. 2024. Elevating Learning Experiences: Leveraging Large Language Models as Student-Facing Assistants in Discussion Forums. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635609

1 INTRODUCTION

In recent years, the enrollment growth in Computer Science (CS) in higher education institutions has outpaced teaching resources [1]. Most important is that the increase in the number of CS educators has not kept pace with student enrollment. Researchers have worked on building tools and examining different ways to increase students' sense of belonging in courses [6, 8]. Thus, our work presents a Natural Language Processing (NLP) solution for assisting teaching staff in answering student questions in large-scale CS courses.

Although Large Language Models (LLM) demonstrate promising domain-specific question-answering capabilities [2, 4, 5], they have several obstacles to their broader application in education. The demands for answer accuracy in education are incredibly high, and hence, LLM hallucination (the tendency for LLMs to generate coherent, yet incorrect responses) is a significant challenge to producing high-quality answers to student questions [9]. One remedy is to fine-tune these models on large amounts of high-quality data of student questions answered by teaching staff. The problem here is that LLM fine-tuning (i) requires immense amounts of computing resources while the alternative of (ii) using publicly hosted LLMs might compromise sensitive student data and invade student privacy.

2 METHODS AND PRELIMINARY RESULTS

The framework for the student-facing and LLM-based discussion forum extension is presented in Figure 1. The framework includes two main modules - (i) the retrieval module for extracting relevant context for the student query and course materials along with (ii) the LLM prompting module to generate the response using an LLM. The framework was tested on a large-scale upper-division data science course (Principles and Techniques of Data Science) at the University of California, Berkeley. The enrollment in the course is around 1,200 students each semester.

2.1 Retrieval Module

First, we demonstrate that student forum Q&A data combined with course documents can be used to aggregate valuable context for assisting teaching staff in answering a student question. We develop a pipeline using non-LLM neural embedding methods to fetch documents and past student Q&A pairs similar to the new student question. This context would then be presented to the teaching staff to answer the question. We perform a thorough human evaluation of the pedagogical benefit of our novel use of unstructured student Q&A data.

Our human evaluation metric is inspired by similar evaluation criteria used in Insta-Reviewer [3]; the final evaluation criteria includes three metrics: *Relevance*, *Diversity*, and *Similarity to Ground Truth*. We evaluate the document retrieval alone and document

*Authors contributed equally to this research.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635609

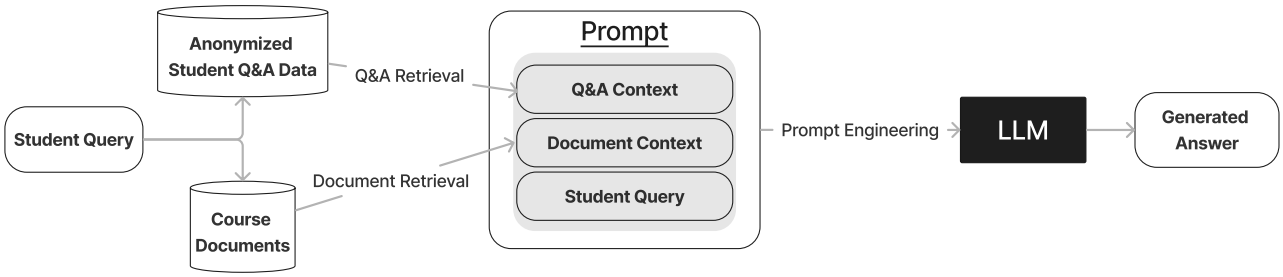


Figure 1: Relevant and broad context for the incoming student query is formed through Q&A and Document retrievals. The context is then prompted with the original student query to an LLM to generate an answer.

and Q&A retrievals combined to quantify the added benefit of unstructured Q&A data in providing relevant and diverse context to the student question used to generate a response similar to an expert. We observe improvements across all three metrics, with 68% of the retrieved context sufficient to replicate the ground truth response. The evaluations were conducted by four of the co-authors of the paper, who are members of teaching staff in Computing courses, and an inter-rater reliability protocol was applied before the start of the evaluations to ensure consistent scoring among the evaluators.

2.2 LLM Prompting Module

The second module leverages the retrieved context from the retrieval module to prompt the LLM to generate an answer. Similar to our retrieval module, we perform human evaluations to demonstrate the value of the retrieved context to the pedagogical quality of the generated response. We also include prompt engineering techniques using expert and subject pre-conditioning.

The evaluation criteria for this stage are also inspired by that of Insta-Reviewer [3]. Since, in this stage, we are evaluating the quality of a generated response rather than the context, we modified the evaluation rubric, and utilized *Relevance*, *Readability*, *Tone*, and *Factuality*. We observe improvements in the results of our framework compared to those of the no-retrieval (only prompt engineering) and zero-shot (only student question) techniques [7].

3 FUTURE WORK

Although we have a working framework that can fetch relevant Q&A pairs and course content and prompt an LLM to respond to student questions, we plan to continue our work to improve, extend, and deploy the framework. We observe that the retrieval component sometimes suffers from a lack of data; this issue can be mitigated by including Q&A data from previous offerings of the course, integrating more sources, including package documentations (for debugging questions) and relevant textbooks (for content questions), and employing more robust data indexing techniques. We believe that the prompting results can be improved by leveraging advanced prompting strategies and possibly fine-tuning the LLM in terms of factuality and employing pedagogical techniques to assist students. That said, we realize that fine-tuning language models can be computationally costly and raise practical issues when setting up the system for a new class.

The framework can be released as a browser extension or as an EdSTEM bot. Before distributing the software for use in the classroom, the system would first be only accessible to the teaching staff. The purpose of this stage is to gather feedback from the course staff to identify any problems and improvements in the system. This would also allow course staff to review answers and make edits, which can subsequently be used for instruction tuning.

Additionally, we plan to evaluate the framework's effectiveness in diverse educational settings, including project-oriented and math-intensive classes, to ensure its applicability to various learning environments.

REFERENCES

[1] Tracy Camp, W. Richards Adrion, Betsy Bizot, Susan Davidson, Mary Hall, Susanne Hambrusch, Ellen Walker, and Stuart Zweben. 2017. Generation CS: the growth of computer science. *ACM Inroads* 8, 2 (2017), 44–50. https://doi.org/10.1145/3084362

[2] Brandon Jaipersaud, Paul Zhang, Jimmy Ba, Andrew Petersen, Lisa Zhang, and Michael R Zhang. 2023. Decomposed Prompting to Answer Questions on a Course Discussion Board. In *International Conference on Artificial Intelligence in Education*. Springer, 218–223.

[3] Qinjin Jia, Mitchell Young, Yunkai Xiao, Jialin Cui, Chengyuan Liu, Parvez Rashid, and Edward Gehringer. 2022. Insta-Reviewer: A Data-Driven Approach for Generating Instant Feedback on Students' Project Reports. *International Educational Data Mining Society* (2022).

[4] Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2023. FlexKBQA: A Flexible LLM-Powered Framework for Few-Shot Knowledge Base Question Answering. *arXiv preprint arXiv:2308.12060* (2023).

[5] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*.

[6] M. Miroyan, S. Weng, R. Shah, L. Yan, and Narges Norouzi. 2024. EIT: Earnest Insight Toolkit for Evaluating Students' Earnestness in Interactive Lecture Participation Exercises. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education*.

[7] Chancharik Mitra, Mihran Miroyan, Rishi Jain, Vedant Kumud, Gireeja Ranade, and Narges Norouzi. 2024. RetLLM-E: Retrieval-Prompt Strategy for Question-Answering on Student Discussion Forums.

[8] Narges Norouzi and Amir Mazaheri. 2023. Context-Aware Analysis of Group Submissions for Group Anomaly Detection and Performance Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[9] Naaz Sibia, Angela Zavaleta Bernuy, Joseph Jay Williams, Michael Liut, and Andrew Petersen. 2023. Student Usage of Q&A Forums: Signs of Discomfort?. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. 33–39.

A Game to Share the Story of LGBTQ+ Pioneers and Influential Computer Scientists

Karina Mochetti
UBC
Vancouver, Canada
mochetti@cs.ubc.ca

ABSTRACT

The lack of diversity in STEM (especially Computer Science) is an important issue to be addressed. Although a lot has been done regarding the gender gap, the same cannot be said for LGBTQ+ representation in the field. This work aims to handle this problem by creating a game that will help to share the story of influential computer scientists who are role models for this minority community within the Computer Science field.

CCS CONCEPTS

• **Social and professional topics** → *Computing education*; *Historical people*; *Sexual orientation*;

KEYWORDS

LGBTQ+ representation in CS, Diversity, Computing education

ACM Reference Format:
Karina Mochetti . 2024. A Game to Share the Story of LGBTQ+ Pioneers and Influential Computer Scientists. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.1145/3626253.3635491

1 INTRODUCTION AND MOTIVATION

The gender gap problem found in STEM, more specifically in the Computer Science field [2, 10] has gathered a lot of attention in the past years. Although the gender gap is indeed an important issue within the area, other minorities, such as People of Color and LBGTQ+, also feel excluded by an environment dominated by (straight) white men [6, 8].

The goal of this work is to develop a game that would illustrate the importance of LGBTQ+ pioneers in the history of Computer Science and encourage people to share that information, highlighting an important minority within the Computer Science field.

Having role models is important for minority groups when choosing and staying in a career [1, 7]. Therefore, telling the story of influential computer scientists who are part of a minority community is an important step when trying to bring diversity and inclusion to the field.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635491

The main inspiration for this work is the *Notable Women in Tech* cards project [4] created to preserve and share the history of women in computing. A deck of cards, containing a short biography about women in the field in which each card, was developed and disclosed.

Although not as extensively researched as the gender gap, the lack of LGBTQ+ representation in the Computer Science field also flows from the overrepresentation of (straight) white men in the field [5, 9]. Like other minority groups, such as women and People of Color, the LGBTQ+ community also suffers from discrimination within the field [3]. Therefore, the need to empower and engage new people from this community in Computer Science is essential and a first step to improving diversity within the field.

Our work's main goal is to create a game focusing on the biography of several LGBTQ+ members in computing. Aside from being part of the LGBTQ+ community, the people on these cards will also be diverse in their countries of origin, generations, topics, and contributions to Computer Science.

2 METHODOLOGY

The game's prototype was developed as a paper-based game. It has two decks of cards, one deck with profile cards and one deck with fact cards. Each player will receive a profile card with a short biography of an LGBTQ+ pioneer in Computer Science, as seen in Figure 1. After reading the profile, the player introduces the name of their pioneer and at each turn, one of the players draws a card from the fact card deck, such as the one seen in Figure 2, and shows it to all players. Each fact card has a small fact that can be found on each profile card, like where that person was born, or which contributions they added to the field. Every player with a profile card that matches the fact drawn will receive the number of points listed on the fact card. For example, if the fact card says *They were born in North America* and *03 points*, everyone who has a profile card with a North American pioneer should raise their hand, say their name, and receive 3 rainbow points. The game ends when a player gets a total of 10 rainbow points. That player is the winner.

The paper-based prototype is ready, but only one copy was created. It does not have a package, and its design is still basic. The next step for this part of the project is improving the design, and producing some copies, so the game can be shared with more people.

We are also working on an online version of the game to be developed as an application for Android and iOS systems. In that version, each player will also have a profile card, and this will be shared with everyone. Instead of a different player drawing fact cards, players will now have to choose among 3 options which fact will be used at each turn. That way, the game forces them to see

which profile card each other player has and the facts related to those people. Since the goal is to maximize the points, the player will try to pick a card that gives them the most points, while giving others fewer points, so knowing if that fact applies to other profile cards will be essential. The online version besides being cheaper to produce, is also easier to share with other universities and Computer Sciences departments not physically close to us.

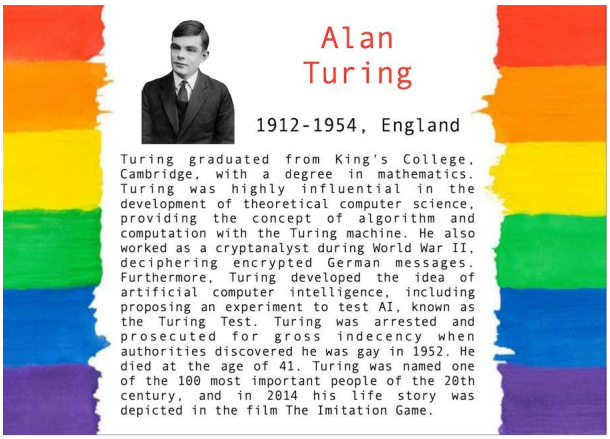


Figure 1: An example of a profile card with a small biography and picture of an LGBTQ+ pioneer in Computer Science.

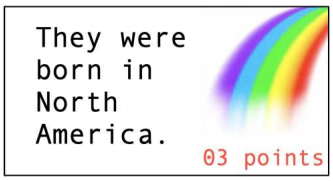


Figure 2: An example of a fact cards with facts found in some biographies.

The game's goal is to give players new role models and people that they can identify with, which is an important step when adding diversity [1, 7]. In the future, we also plan to create a website where people can submit other biographies and profile pictures to extend the profile cards. At the moment we have only 12 people who were selected based on their profile found online. We looked for people that nowadays are open about being part of the LGBTQ+ community, or who faced challenges for being part of this minority in the early days of Computer Science. Those pioneers were chosen to include different genders, nationalities, generations, and research fields. The list of selected names is:

- Alan Turing, 1912-1954, England
- Peter Landin, 1930-2009, England
- Tim Gill, 1953-, USA
- Luca Trevisan, 1971-, Italy
- Edith Windsor, 1929-2017, USA
- Sophie Wilson, 1957-, England
- Audrey Tang, 1981-, Taiwan
- Lynn Conway, 1932-, USA
- Jon 'maddog' Hall, 1950-, USA
- Christopher Strachey, 1916-1975, England
- Mary Ann Horton, 1955-, USA
- Sofia Kovalevskaya, 1850-1891, Russia

3 RESULTS AND IMPACT

The prototype of the paper version is ready, now an application based on the game is under development. The paper version has been tested in events organized by the LGBTQ+ group at our university, and having an application will make it easier to spread it to other universities as well. That way we want to share knowledge of amazing people and their contribution to Computer Science and also invite others to share their knowledge as well, by adding new bio cards to the game.

For our poster presentation, we envision not only having a poster describing our game but having some paper-versions ready to people test it and give us informal feedback.

It is important to note that in contrast to other minority groups, such as women and People of Color, it is harder to find people who are openly LGBTQ+. With this project, therefore, I not only want to share their impact in the field but also invite others to feel free and comfortable to open up about this personal side of their life, if they wish to, which they should be proud of.

REFERENCES

[1] Elizabeth M. Almquist and Shirley S. Angrist. 1971. Role model influences on college women's career aspirations. *Merrill-Palmer Quarterly of Behavior and Development* 17, 3 (1971), 263–279.
[2] David N Beede, Tiffany A Julian, David Langdon, George McKittrick, Beethika Khan, and Mark E Doms. 2011. Women in STEM: A gender gap to innovation. *Economics and Statistics Administration Issue Brief* 04-11 (2011).
[3] Carlos Alberto Damas and Karina Mochetti. 2019. An analysis of homophobia on vandalism at Wikipedia.
[4] Katy Dickinson, Susan Rogers, and Jessica Dickinson Goodman. 2014. "Notable Women in Computing" poster and playing cards.
[5] Amanda Menier, Rebecca Zarch, and Stacey Sexton. 2021. Broadening Gender in Computing for Transgender and Nonbinary Learners. In *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. IEEE, 1–5.
[6] Ryan A Miller and Megan Downey. 2020. Examining the STEM Climate for Queer Students with Disabilities. *Journal of Postsecondary Education and Disability* 33, 2 (2020), 169–181.
[7] Catherine Porter and Danila Serra. 2020. Gender differences in the choice of major: The importance of female role models. *American Economic Journal: Applied Economics* 12, 3 (2020), 226–54.
[8] Katherine Rainey, Melissa Dancy, Roslyn Mickelson, Elizabeth Stearns, and Stephanie Moller. 2018. Race and gender differences in how sense of belonging influences decisions to major in STEM. *International journal of STEM education* 5, 1 (2018), 1–14.
[9] Jane G Stout and Heather M Wright. 2016. Lesbian, gay, bisexual, transgender, and queer students' sense of belonging in computing: An Intersectional approach. *Computing in Science & Engineering* 18, 3 (2016), 24–30.
[10] Jennifer Tsan, Kristy Elizabeth Boyer, and Collin F Lynch. 2016. How early does the CS gender gap emerge? A study of collaborative problem solving in 5th grade computer science. In *Proceedings of the 47th ACM technical symposium on computing science education*. 388–393.

Advice for the First Time Hardware CTF Organizer

Chad Mourning
mourning@ohio.edu
Ohio University
Athens, Ohio, USA

ABSTRACT

In 2022, the IEEE Computer Society provided funding for a hardware-security based Capture the Flag competition. This paper reports my experiences orchestrating the CTF and draws comparisons to two software-security based CTFs at the same institution in adjacent semesters. Much of the literature in this area is focused on selecting problems to enhance the skills of the attendees or generate outcomes that persist. In contrast, this paper is intended to provide tips, warnings, and other general advice for first time, or niche CTF organizers.

ACM Reference Format:
Chad Mourning. 2024. Advice for the First Time Hardware CTF Organizer. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635582

1 INTRODUCTION

Cybersecurity Catpure-The-Flag (CTF) competitions are a popular category of interactive, hands-on learning exercises. CTFs were first performed at DEFCON in 1996; the popular CTF leaderboard CTFTime lists 269 CTFs that occurred in 2022.

There are two sub-categories of CTF: Jeopardy-style, and Attack-Defend, sometimes referred to as Red Team-Blue Team, named after the information security (InfoSec) color wheel. Jeopardy-style contests are easier to organize, easier to scale in difficulty, and the vast majority of contests.

Traditional software-centric CTFs focus on challenges involving, for instance: attacking a binary file, through reverse engineering, attacking a remote application, or using knowledge of cryptography to reveal the flag. Hardware CTFs, the focus on this paper, rely on finding exploits in the hardware underneath by, for instance: performing a side-channel analysis [5], de-soldering components or inserting your own, or voltage glitching [2]. [7] lays out four rules for what differentiates a software from a hardware challenge.

This paper is intended as a helpful guide for first time hardware CTF organizers. It includes advice on aspects of event planning that the author anticipated before running their event, as well as a section on unexpected problems encountered.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635582

2 RELATED WORK

Much of the literature regarding CTFs is on assessment of outcomes and design of exercises there is very little on the practical considerations in running the actual competition. [6] provides a guide on organizing a hackathon; there is significant overlap between hackathons, broadly, and CTFs in particular. [3] describes the difference between *developing* a CTF and *running* one, but lacks many practical details, such as “where will people park?" [8] presents an overview based on the experiences of eight CTFs at Lawrence Livermore National Laboratory and 36 contemporary CTFs. It breaks down the costs, taxonomizing hardware and human costs as well as recurring versus nonrecurring; however, the “administrative costs" listed are only for the development of the contest, and do not include any facility costs.

Running a conventional CTF continues to become easier as many open-source frameworks [4] exist to facilitate contests. You may notice a theme in some of these papers: virtual environments,– one striking difference between traditional software CTFs and hardware CTFs is the need to be in person to hack the hardware. [1] is an survey paper from 2023, which provides a taxonomy of CTFs as the authors see it. To highlight the rarity of Hardware CTFs, this taxonomy has no mention of hardware-centric CTFs.

3 HARDWARE SECURITY

Cybersecurity education is integrated into most CS programs in some fashion as a requirement of ABET’s *General Criterion 5.* Historically, a heavy focus of Cybersecurity has been on networks; then the focus shifted to applications; but there has been a growing focus on securing the hardware the software runs on.

The 2017 ACM Cybersercurity Curricula is a recommendation of topics covered when teaching cyberscurity divided into eight *Knowledge Areas.* Knowledge Area three, “Component Security," includes the topics covered in what is colloquially known as Hardware Security.

There is no reason these skills cannot also be tested in a CTF, although they are more resource intensive than software CTFs. [7] provides a list of a few recurring, or previously recurring, hardware CTFs; Hardwear.io, for instance, has hosted an annual Hardware CTF in The Netherlands since 2017. Software focused CTFs sometimes include at least one hardware challenge such as Google CTF still recurring as of 2023.

4 THE CONTESTS

In this paper I discuss 3 events at Ohio University (OHIO): **CTF1** - Software Spring ’22, **CTF2** - Hardware Fall ’22, and **CTF3** - Software Spring of ’23. In 2022, I was funded by the IEEE Computer Society to run a hardware-security themed CTF. Expected costs included: swag, meals, support for student org assistance, competition hardware such as the HaHa Board [9], and travel support.

Half the budget was for travel support for participants from other schools. There are 14 public 4-year universities in the state of Ohio, and a handful of private universities of size that I thought might be interested; 20 travel grants of $750 were requested.

CTF2 in Context There were only 6 competitors for **CTF2**. This is compared to 13 for **CTF1**, with similar advertising and no travel support, and 65 for **CTF3**, with lessons learned from **CTF2** and co-located events. For reference, **CTF3** was a wild success for a 24 hour event at OHIO; a Major League Hacking sponsored hackathon the previous month attracted 30. This demonstrates the complete ineffectiveness of the travel grants in isolation; Ohio State University is a 90 minute drive from OHIO, and no competitors were interested in driving down and back the day of the competition for the $750.

The Expected

Save The Date Perhaps the most important aspect in planning a multi-university event is the date. There is a CTF occurring every weekend; many support virtual attendance which is not as likely an option for hardware CTFs. Also consider non-CTF events that may target the same audience. When planning **CTF2**, students from Ohio Northern University expressed interest in participating in the hardware CTF, but there was a robotics contest the same weekend.

Parking If the reader's organization is similar to the author's, Parking Services may be the least popular unit on campus. At OHIO, the largest consumer of parking is athletic events; I recommend planning your event for when there are no home games.

Catering OHIO has a policy that any event that spends over $250 on food must use university catering or seek an exemption.

Advertising My largest advertising blunder was focusing on reaching out to the Department Chairs at the target universities. Of 24 Chairs I reached out to, 3 responded; I recommend to reach out to student orgs instead. Chapters of ACM/IEEE first, but many schools have separate cybersecurity clubs. Also search for university faculty with research interests in Cybersecurity to contact directly.

Sponsorship **CTF1** had a sponsor of JP Morgan Chase. **CTF2** was covered by the funds from the IEEE Computer Society. **CTF3** had a combination of sponsors for the prize pool, and most administrative costs were covered by funding from the Russ College of Engineering and Technology.

Swag For t-shirts, OHIO encourages event planners to use their own printing services; prices were reasonable. Many universities are strict with branding; I would recommend branding with the event name, not the logo of the school.

Registration The flexibility of same-day registration is beneficial; headcount from pre-registration is useful when estimating catering and swag needed. To incentivize early registration, the strategy for **CTF3** was to provide limited travel support in order of registration.

Content The problems for **CTF2** were drawn from the labs for our Hardware Security course.

The Unexpected

Room Reservations Revisited At OHIO, room reservations are not binding, particularly classroom spaces. My recommendation is that for first time events is to over-reserve, just in case.

Physical Security CTFs and Hackathons are frequently 24 hour+ events. OHIO has a policy that all overnight events must cover the cost of an extra shift for our campus police.

Custodial At OHIO it is in the custodial union's contract that there are additional charges for custodial service after events.

The Weather One team excited to attend **CTF2**, from the University of Cincinnati, were snowed in. Hardware focused CTFs are less likely to support virtual attendance.

Reproducability I expected that it should be straightforward to migrate labs from our Hardware Security course to challenges for the CTF. Unfortunately, the results produced by the hardware we were using were not always reproducible.

Supply Chain Issues Since this is a hardware-centric CTF, you may need to purchase new hardware for the competition. For **CTF2**, I planned to accommodate 20 teams. One piece of equipment I required had a limit of 2 per month and was backordered from many vendors; for a contest of twenty this would require a 10 month lead time to acquire the components.

5 FINAL RECOMMENDATIONS

The hardware CTF ecosystem is not mature yet; organizers cannot rely on the energetic pool of existing competitors seeking new competitions as exists in the software CTF space. Most undergraduate Computer Science students do not have the skills or inclination to do hardware security, Computer Engineers are the target audience for Hardware CTFs. Start with parity: host a CTF that is half and half. Many competitions include one hardware problem, so this will need ratcheted up over time. Finally, when fostering a new ecosystem, make sure to include a few easy problems. First time organizers should not discourage first time competitors who could become repeat participants.

ACKNOWLEDGMENTS

This work was funded in part by the IEEE Computer Society, as well as the Air Force Research Lab's project FA8650-20-2-1136.

REFERENCES

[1] Tyler Balon and Ibrahim Baggili. 2023. Cybercompetitions: A survey of competitions, tools, and systems to support cybersecurity education. *Education and Information Technologies* (2023), 1–33.

[2] Claudio Bozzato, Riccardo Focardi, and Francesco Palmarini. 2019. Shaping the glitch: optimizing voltage fault injection attacks. *IACR transactions on cryptographic hardware and embedded systems* (2019), 199–224.

[3] Nicholas Childers, Bryce Boe, Lorenzo Cavallaro, Ludovico Cavedon, Marco Cova, Manuel Egele, and Giovanni Vigna. 2010. Organizing large scale hacking competitions. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 7th International Conference, DIMVA 2010, Bonn, Germany, July 8-9, 2010. Proceedings 7*. Springer, 132–152.

[4] Stela Kucek and Maria Leitner. 2020. An empirical survey of functions and configurations of open-source capture the flag (ctf) environments. *Journal of Network and Computer Applications* 151 (2020), 102470.

[5] Thanh-Ha Le, Cécile Canovas, and Jessy Clédiere. 2008. An overview of side channel analysis attacks. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*. 33–43.

[6] Ei Pa Pa Pe-Than, Alexander Nolte, Anna Filippova, Christian Bird, Steve Scallen, and James D Herbsleb. 2018. Designing corporate hackathons with a purpose: The future of software development. *Ieee Software* 36, 1 (2018), 15–22.

[7] Paolo Prinetto, Gianluca Roascio, and Antonio Varriale. 2020. Hardware-based capture-the-flag challenges. In *2020 IEEE East-West Design & Test Symposium (EWDTS)*. IEEE, 1–8.

[8] Clark Taylor, Pablo Arias, Jim Klopchic, Celeste Matarazzo, and Evi Dube. 2017. {CTF}:{State-of-the-Art} and Building the Next Generation. In *2017 USENIX Workshop on Advances in Security Education (ASE 17)*.

[9] Shuo Yang, Shubhra Deb Paul, and Swarup Bhunia. 2021. Hands-On Learning of Hardware and Systems Security. *Advances in Engineering Education* 9, 2 (2021).

Implementation and Evaluation of Equitable Practices in an Introductory CS Course

Nadia Najjar
Software and Information Systems
University of North Carolina at Charlotte
nanajjar@charlotte.edu

Debarati Basu
Engineering Fundamentals
Embry-Riddle Aeronautical University
basud@erau.edu

ABSTRACT

Lack of diversity is a major challenge in CS education. One way to broaden participation and increase student success is the adoption of equitable pedagogical and grading practices. Lack of evaluation of such practices within the context of undergraduate CS education results in less adoption. This study is the first step in a broader project examining such practices and their impact on undergraduate CS students. Factors such as confidence and sense-of-belonging have been linked to increase students' success. In this study, we report on the evaluation of four practices commonly adopted in pursuit of equity, in terms of students' perceptions and how they impact students' sense-of-belonging, confidence, and performance in a computer science course. We report on students' feedback from a CS2 course in the Spring 2023 semester. Our results show that students valued these practices and perceived them as having a positive impact on their course experience in relation to these constructs. Students also reported that these practices gave them better control of their learning, allowed them to learn from their mistakes, and have a better understanding of the course content.

KEYWORDS

Equitable Practices, CS education, Undergraduate education

ACM Reference Format:
Nadia Najjar and Debarati Basu. 2024. Implementation and Evaluation of Equitable Practices in an Introductory CS Course . In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635590

1 INTRODUCTION

In recent decades, the computer science (CS) education community has been working to enhance diversity to better reflect the student body on US campuses and society as a whole. A key strategy to achieve this diversity is by fostering equity in our classrooms. Various teaching and grading methods have proven effective in promoting equity among students, ensuring similar learning outcomes regardless of systematic disparities, hardships, underrepresentation, or disabilities. Our previous study on faculty's views on equity in CS education revealed their recognition of its importance but also their uncertainty about its exact meaning, implementation and impact [1, 7]. We identified four equitable practices adapted from K-12 literature [4, 6, 8], which were regularly used by 21% to 60% of faculty participants in our previous study. We implemented these practices in an introductory CS2 course at a doctoral university with high research activity in the south-eastern US during the Spring of 2023.

The four practices are as follows (short name in parenthesis):

- Submit assignments without penalty after the instructor alerts that the deadline has passed (Allow late submission)
- Resubmit an assignment after the instructor's feedback to improve performance (Allow resubmission)
- Attend a review session before an exam (Conduct review session)
- Meet with the instructional team after the early and/or the midterm alerts (Schedule meeting)

This pilot study intentionally assesses the impact of these four practices on students' perceptions, including their sense-of-belonging, confidence, and performance in the course. These practices are adopted to enhance participation and offer avenues for enhancing students' performance in the course specifically and, more broadly, the CS field. The aim is to furnish CS educators with a more comprehensive evaluation of these practices as they seek to promote equity in their classrooms.

2 LITERATURE REVIEW

We utilize the following definition of equity [7]:

> ***Equity in CS Education*** *is the absence of systematic disparities in educational outcomes between social groups who have different levels of underlying social advantage/disadvantage.*

This definition acknowledges the existence of systematic disparities and emphasizes that the goal of equitable education is to eliminate these disparities with respect to **educational outcomes**, which may include passing grades, course completion, or graduation rates. It underscores the importance of achieving uniform learning outcomes across all groups and recognizes that varying levels of underlying social advantage can influence these outcomes, potentially requiring additional resources or special attention for some groups to achieve similar educational outcomes as others.

Previous work highlights various effective equity-promoting practices such as individualized teaching plans based on student progress, avoiding behavior-based grading, and using meaningful context, scaffolding, peer collaboration, and multiple forms of expression. While many of these practices are typical in K-12 education we adapted some for use in undergraduate CS education [2, 4, 8]. Studies reveal that the student learning experience and

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635590

success can be enhanced when educational settings are structured to foster a sense-of-belonging and support student self-efficacy. Such settings encourage students to utilize resources available to them and enhance their success [9].

3 METHOD

We conducted an IRB-approved study and utilized a survey in a CS2 classroom to collect students' perceptions of these practices and the impact they had on their sense-of-belonging, confidence, and performance. The survey included demographic questions and asked students to rate the following statements in relation to each of the practices using a 5-point Agree/Disagree Likert scale:

- Helps me improve my performance in the course
- Helps me improve my confidence in the course
- Contributes to my feeling of being accepted in the course
- Contributes to my feeling of being comfortable in the course
- Contributes to my feeling of being supported in the course

The last three statements on sense-of-belonging were adapted from [5]. The survey also included open-ended questions for students to share their thoughts on how these practices impacted their experience and a more general question asking if there were any other practices that they think have impacted their experience in the course. The survey was shared with students after they completed the final exam. As a result, we had 11 participants who completed and consented to the study out of 69 students in the class.

4 RESULTS

Among the participants, 64% (7/11) utilize financial aid, 55% (6/11) work part-time, three were females, seven were males and one did not disclose their gender. In terms of race, four were whites, three were Black or African American, three were Asians and one preferred not to answer.

Figure 1 shows how the participants rated each of the four practices in terms of the three constructs, performance, confidence, and sense-of-belonging. Overall, our results show that participants found all the practices had a positive impact on their performance, confidence, and sense-of-belonging in the course (average rating more than 4). However, 'schedule meeting' where students get an opportunity to meet with the instructor to make a catch-up plan seemed to be the most effective practice contributing to each of the three constructs. Additionally, contrary to our expectation, we observed that 'allow late submission', where students get an opportunity to submit assignments after the instructor alerts them that the deadline has passed, had the least effect on these constructs.

Students' responses to the open-ended questions were analyzed using thematic analysis [3] to find out what students thought about each of the practices. The analysis revealed four main themes: confidence, performance, sense-of-belonging, and autonomy. Three of the themes directly relate to the targeted constructs of our study. Students valued the ability to resubmit work as it allowed them to learn from failures. P1 said "I loved being able to do this. I hate when you turn something in, receive little to no feedback...". Allowing late submissions gave students the opportunity to prioritize and balance between school work and other commitments. Sense-of-belonging was also highlighted in students' responses. P3 said "Being able to feel seen, be included (despite being another major in another

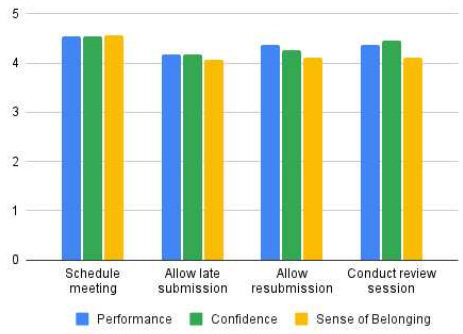


Figure 1: Students' Perceptions about the Equitable Practices

department) made me feel equal to my peers and therefore have a chance." The fourth theme that we identified was that students valued autonomy. The ability for them to work at their own pace and prioritize allowed them to focus on learning and gave them a better understanding of the content.

5 CONCLUSIONS AND FUTURE WORK

This study serves as a preliminary exploration laying the groundwork for a comprehensive large-scale evaluation of equitable pedagogical practices. Our primary objective is to gain insights into students' perceptions regarding the impact of these practices on their academic performance, confidence, and sense-of-belonging. Our findings revealed that students highly valued these pedagogical approaches as they facilitated their ability to concentrate on grasping key concepts and learn from their mistakes. Furthermore, they reported an enhanced sense of control, enabling them to prioritize and take ownership of their learning. One limitation of this study is the low number of participants as participants were recruited after the final exam of the course. In our future work, we will conduct the survey before the final exam within class time to address this.

REFERENCES

[1] D. Basu and M. Pérez-Quiñones. 2023. Computing Faculty's Definition of Equity and Their Adoption of Equitable Practices. In *RESPECT 2023*. IEEE Press.
[2] J. Feldman. 2019. *Grading for equity: What it is, why it matters, and how it can transform schools and classrooms.* Corwin Press.
[3] J. Green, K. Willis, E. Hughes, R. Small, N. Welch, L. Gibbs, and J. Daly. 2007. Generating best evidence from qualitative research: the role of data analysis. *Australian and New Zealand journal of public health* (2007).
[4] Thomas R Guskey. 2000. Grading policies that work against standards... and how to fix them. *Nassp Bulletin* 84, 620 (2000), 20–29.
[5] S. Krause-Levy, W. Griswold, L. Porter, and C. Alvarado. [n.d.]. The relationship between sense of belonging and student outcomes in CS1 and beyond. In *17th ACM Conference on International Computing Education Research*.
[6] S. McGee, R. McGee-Tekula, J. Duck, C. McGee, L. Dettori, R. Greenberg, E. Snow, D. Rutstein, D. Reed, B. Wilkerson, et al. 2018. Equal outcomes 4 all: A study of student learning in ECS. In *49th ACM SIGCSE*.
[7] M. Pérez-Qui nones and D. Basu. 2023. Equity in Computing Education is Not a Political Agenda. In *Proceedings of the 2023 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. IEEE Press, 3.
[8] N. Shah, Co. Lewis, R. Caires, N. Khan, A. Qureshi, D. Ehsanipour, and N. Gupta. 2013. Building equitable computer science classrooms: Elements of a teaching approach. In *44th ACM SIGCSE*. 263–268.
[9] D. S Yeager, P. Hanselman, G. Walton, J. Murray, R. Crosnoe, C. Muller, E. Tipton, B. Schneider, C. Hulleman, C. Hinojosa, et al. 2019. A national experiment reveals where a growth mindset improves achievement. *Nature* 573, 7774 (2019), 364–369.

Incorporating Technical Interview Training into an HBCU Pre-Freshman Programming Bootcamp

Whitney Nelson†
Broadening Participation in Computing
Morehouse College
Atlanta GA, USA
whitney.nelson@morehouse.edu

Kinnis Gosha
Broadening Participation in Computing
Morehouse College
Atlanta GA, USA
kinnis.gosha@morehouse.edu

Talitha Washington
AUC Data Science Initiative
Atlanta University Center Consortium
Atlanta GA, USA
twashington@aucenter.edu

Nina Gilbert
The Morehouse Center for Excellence in Education
Morehouse College
Atlanta GA, USA
nina.gilbert@morehouse.edu

ABSTRACT

Internship experiences allow students to authentically participate in industry while still obtaining mentorship and development opportunities. The Morehouse Center for Broadening Participation in Computing, in collaboration with the Atlanta University Center (AUC) Data Science Initiative, Lockheed Martin, and Amazon Photos joined forces to expand a PreFreshmen eXperience (PFX) program that supports student exposure to programming and interviewing skills. Specifically, incoming Computer Science and Software Engineering freshmen at Morehouse College, Spelman College, and Clark Atlanta University were eligible to apply to the program. The goal of this program was to not only prepare students for courses but also expose them to technical interviewing skills, resources, tools, and experiences in hopes that they will use this information to apply to an internship following their freshman year.

ACM Reference format:

Whitney Nelson, Kinnis Gosha, Talitha Washington, & Nina Gilbert. 2024. Incorporating Technical Interview Training into an HBCU Pre-Freshman Programming Bootcamp. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V.2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635564

1 INTRODUCTION

Limited coding skills and minimal exposure to expected technical interviewing behaviors serve as a barrier for any student to obtain summer internships. According to a 2016 Google report, Black students are less likely than white students to have classes dedicated to computer science at the high school they attend suggesting that many students have no prior coding knowledge before college. While Silicon Valley thrives each summer by hosting interns at various tech companies, less than 1% of coders are Black. Although several initiatives have been created to provide early exposure to Black students interested in computing; retention is still an issue for academic programs and entering the computing workforce can be difficult for many students. Internships have been cited to help students decide rather they would stick to their computing major [8] and enter the workforce. Research suggest that supplementary training is required to help students successfully launch their computing careers.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03. https://doi.org/10.1145/3626253.3635564

2 PROGRAM DESIGN

Coding experience is critical for the major and academic performance of incoming students due to retention trends in their first three coding courses (Programming I, Programming II, and Data Structures). According to the Morehouse 2022 factbook, 2021 had 90 CS/SWE majors at the freshman level, 43 at the sophomore level, and only 24 graduates. The goal of this program is to increase the retention of incoming Morehouse and AUC CS/SWE majors and boost confidence to apply for internships. We chose to expose students to Python as this is a beginner friendly coding language. Instruction began the first day where students were put in groups to meet their peers and solve computation thinking problems. As the classroom was flipped, students completed learning modules on Courseara introducing them to computing concepts. These were coupled with live lectures and code walk throughs to compliment the material learned outside of the virtual classroom. Core Python learning objectives of the boot camp include the following: basic python syntax, arithmetic operators, variables and expressions, conditional code, functions, loops and iterations, strings, lists, and dictionaries. Students also completed four individual curriculum assignments applying the learning objectives, participated in two peer programming coding challenges, completed code reviews, and worked on a group project where they were prompted to build a Python game.

Students were awarded a certification of completion and a $800 stipend aimed to go towards a new laptop to support their freshman technological needs.

2.1 Program Format

Students must have been admitted and enrolled into an AUC institution for Fall 2023 semester. The program was held virtually to accommodate the by-costal participants as most students would not be on campus until the fall. The bootcamp lasted four weeks over the summer and with each session tentatively schedule for half the day on weekdays. Daily stand-ups were from Monday to Friday were held at 1 PM EST on Zoom. Office hours were held daily from 4:00 PM and remained open for one hour. Learning and assignments were completed asynchronously. In addition to teaching the academic topics, engineers and business leaders from Lockheed Martin, Amazon Photos, and Karat were invited to give talks with students to ensure they are knowledgeable on what it takes to prepare themselves for a computing career. There were two live guest speaker Zoom sessions with each session lasting approximately one hour. A representative from Lockheed Martin spoke on internships, time management, and career success. Several representatives from Amazon Photos discussed interview and resume tips. A representative from Karat provided a video of where students learned about technical interviewing techniques.

2.2 Technical Interview Preparation

Technical interviews are used to gauge a candidate's technical, interpersonal, and verbal skills for a job position [4]. Pre-exposure to technical interview techniques such as whiteboarding results in favorable psychological behaviors [1] and are necessary practice to hone in on computing skills [5]. Alongside the course modules and assignments, students used Hakerrank and Karat as a tool to practice coding interview problems. Hackerrank is an online coding practice platform where individuals can create profiles and gain points, badges, and awards for completing coding challenges. Each student built profiles on Hackerrank where they completed several challenges to obtain badges that can be shared on their resumes and with recruiters. Students were also required to complete one technical interview with a professional from Karat's Brilliant Black Minds Program, an online network for Black engineers to enter the tech industry. By completing this interview, students were automatically signed up for the program where they will receive interview practice resources and job opportunities through the platform.

3 EVALUATIONS AND RESULTS

Forty students signed up for the program, thirty-two students accepted the offer, and twenty-nine participants completed all sessions of the virtual four-week boot camp exposing them to computing concepts before they begin their journey in the Fall. A presurvey was distributed to participants before the program to assess their knowledge of the core curriculum and exposure to technical interview practice. Only one student reported completing a technical interview prior to the boot camp. About 70% had one year or less of programming experience. An additional survey was taken at the end of the program where 78% of program participants reported that they plan to continue to pursue a degree in computing: "I had no experience in computing. I am now comfortable with pursuing a major in computer science…I feel as if I am equipped to start my college career." Students with programming experience expressed gaining a broader coding foundation since complete the program: "This program has helped me learn more about coding than I did when I first coded in the 11th grade for the first semester." Three fourths of the program participants plan to apply for an internship the following their freshman year with 50% planning to use Karat as a platform to practice technical interviewing. Over 80% said that they were likely to recommend this program to an incoming AUC student. An additional survey will be distributed at the end of their freshman year aimed at tracking their progress with technical interviews and obtaining an internship.

4 CONCLUSIONS

Students spent each week attending daily stand-ups, completing online modules, working on assignments, attending live lectures, and participating in group projects. The students learned basic python programming concepts. Students were also exposed to technical interview tools like Hackerrank and Karat. Survey results showed that most students were unaware of technical interviews. Through this program, each student was able to complete at least one technical interview providing them with early exposure to the process. Guest speakers from Lockheed Martin and Amazon Photos also spoke to the rising freshmen about applying to internships, career success, and resume tips, and provided interview suggestions.

ACKNOWLEDGMENTS

We would like to thank the AUC Data Science initiative.

REFERENCES

[1] Dillon, E., Williams, B., Ajayi, A., Bright, Z., Kimble-Brown, Q., Rogers, C., ... & Williams, K. L. (2021, May). Exposing Early CS Majors to Coding Interview Practices: An HBCU Case Study. In 2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT) (pp. 1-4). IEEE.

[2] Diversity gaps in computer science -google search. (n.d.). https://services.google.com/fh/files/misc/diversity-gaps-in-computer-science-report.pdf

[3] Factbook. Morehouse. (2023, September11).https://morehouse.edu/about/services-and-administration/data-analytics-institutional-research-and-effectiveness/fact-book/

[4] Ford, D., Barik, T., Rand-Pickett, L., & Parnin, C. (2017, May). The techtalk balance: what technical interviewers expect from technical candidates. In 2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE) (pp. 43-48). IEEE.

[5] G. L. McDowell, Cracking the coding interview: 189 Programming Questions and Solutions. CareerCup, 2015, ISBN: 9780984782857.

[6] Hall Jr, P., & Gosha, K. (2018, June). The effects of anxiety and preparation on performance in technical interviews for hbcu computer science majors. In Proceedings of the 2018 ACM SIGMIS Conference on Computers and People Research (pp. 64-69).

[7] Kapoor, A., & Gardner-McCune, C. (2020, February). Barriers to securing industry internships in computing. In Proceedings of the Twenty-Second Australasian Computing Education Conference (pp. 142-151).

[8] Newman, Christopher B. "Corporate Internships, Undergraduate Research, and Finances: Successful African American Engineers€™ Consideration of Immediate Workforce Entry or Graduate School." Journal of STEM Education: Innovations and Research 21.1 (2020).

Use of Large Language Models for Extracting Knowledge Components in CS1 Programming Exercises

Rose Niousha
University of California, Berkeley
Berkeley, USA
rose.n@berkeley.edu

Muntasir Hoq
North Carolina State University
Raleigh, USA
mhoq@ncsu.edu

Bita Akram
North Carolina State University
Raleigh, USA
bakram@ncsu.edu

Narges Norouzi
University of California, Berkeley
Berkeley, USA
norouzi@berkeley.edu

ABSTRACT

This study utilizes large language models to extract foundational programming concepts in programming assignments in a CS1 course. We seek to answer the following research questions: **RQ1.** How effectively can large language models identify knowledge components in a CS1 course from programming assignments? **RQ2.** Can large language models be used to extract program-level knowledge components, and how can the information be used to identify students' misconceptions? Preliminary results demonstrated a high similarity between course-level knowledge components retrieved from a large language model and that of an expert-generated list.

ACM Reference Format:
Rose Niousha, Muntasir Hoq, Bita Akram, and Narges Norouzi. 2024. Use of Large Language Models for Extracting Knowledge Components in CS1 Programming Exercises. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* , 2 pages. https://doi.org/10.1145/3626253.3635592

1 INTRODUCTION AND BACKGROUND

An essential part of designing an inclusive Computing curriculum is to identify essential topics that students will master. Educators create lesson plans and design homework assignments based on the breakdown of topics that must be covered in each course component. The notion of course topics, also known as Knowledge Components (KCs), is introduced within the Knowledge-Learning-Instruction (KLI) framework [3]. KCs represent the competencies students develop through programming exercises, offering profound insights into learning and demonstrating achievement of learning outcomes. The understanding of KCs empowers educators to create more effective CS curricula. However, educators find it challenging to identify which KCs should be part of the curriculum and to what extent KCs are assessed in course assessment elements.

Recent advances in Artificial Intelligence (AI), particularly the emergence of Large Language Models (LLMs), opened up a new avenue in Computing education research. Researchers have used LLMs for generating course materials [6], program repair [4], and IDE-integrated chatbots [5, 8]. LLMs have the potential to facilitate curriculum development for educators and introduce innovative teaching methods. In this work, we explore using LLMs to extract KCs in programming assignments of a CS1 course to facilitate educators' ability to evaluate the coverage of their designed curriculum. We plan to extend the work to extract KCs present in student programs and tie missing expected KCs to misconceptions.

Prior work by Shi et al., KC-Finder, is an automated KC discovery system to uncover KCs within a CS1 course in Java [7]. Shi et al. demonstrated that KC-Finder successfully discovered KC candidates, and students demonstrated improved mastery of the KCs over time. However, the discovered KCs by KC-Finder lacked clearly defined expert concepts and required substantial training data. We address both issues by 1) automatically extracting KCs using LLMs and 2) evaluating the effectiveness of KCs extracted through LLM prompting compared to an expert ground-truth KC list. This study can potentially utilize KC extraction using LLMs to identify student misconceptions and provide personalized feedback.

2 METHODOLOGY AND PRELIMINARY RESULTS

We used a Java programming dataset collected from a CS1 course from the Spring 2019 semester of a public university in the US using the CodeWorkout platform [1]. The course had 50 programming exercises (5 assignments with 10 programming exercises in each). We randomly selected one correct student submission for each problem. We then extracted the problem statements for the corresponding programs. We considered correct submissions since incorrect student programs may lack an associated KC. Next, we prompted two LLMs (Llama2[1] and GPT3.5[2]) with three scenarios:

I. Both Student Program and Problem Description.
II. Only Student Program.
III. Only Problem Description.

We performed prompt engineering and identified the prompt most helpful in extracting KCs. This prompt, in case only the program is passed to LLM, is: *"List ONLY the Knowledge Component*

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635592

[1] https://chat.lmsys.org/?arena
[2] https://chat.openai.com/

name(s) for this code.", and is modified to refer to the description or both description and code in different ablation scenarios.

We aggregated the extracted KCs and removed semantic duplicates to obtain a final list of KCs for the course. Since there were three ablation scenarios and two types of LLMs, the process outlined in Figure 1 is repeated six times, resulting in six KC lists. Two co-authors (CS1 instructors) labeled the dataset with the presented KCs used as the ground-truth data. They first mutually labeled 10% of the data to ensure consistency. Then, they divided the rest of the data and labeled them individually. Due to the cumulative nature of CS1, the importance of KCs dynamically changes over time, and some KCs go into the background (become passive KCs) as they are no longer the point of practice in the new problem. Thus, experts only labeled seemingly active KCs for each problem.

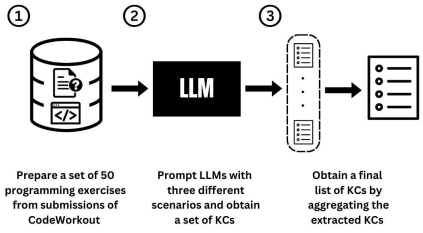


Figure 1: Process of Extracting the List of KCs using LLMs.

2.1 Impact of the Prompt

To assess the level of resemblance between the ground truth and six KC lists, we calculated each Jaccard index.

I. Prompt with student program and problem description. The Jaccardrd index for Llama2 and GPT3.5 were 67% and 83%, respectively.
II. Prompt with only the student program: The Jaccard index for Llama2 and GPT3.5 were 54% and 67%, respectively.
III. Prompt with only problem description: The Jaccard index for Llama2 and GPT3.5 were 28% and 82%, respectively.

Overall, the results show that the KCs generated by LLMs most closely resemble ground truth when prompted with student program and problem description. However, in the third scenario, where GPT3.5 was prompted with only the problem description, the Jaccard index was very close to that of the first scenario. This suggests that GPT3.5 when relying solely on the problem description, can yield responses with a level of similarity comparable to the ideal case, which not only conserves tokens but also holds the potential to reduce the associated cost of automation.

There were minor omissions in the LLMs compared to the ground truth. For instance, the LLMs did not capture the KC "Chain Conditionals" in the ground truth in any scenarios. Moreover, "Nested Loops" were only captured by LLMs in the first scenario, but in the other two scenarios, LLMs only captured "Loops" without specifying the type of loops.

In the third scenario, Llama2 generated unrelated components that were not KCs. Such components include "Lottery Ticket," "Restaurant Table Management," "Temperature and Season,", taken from the scenario described in the question.

2.2 Impact of the LLM

GPT3.5 consistently achieved a higher Jaccard index than Llama2. This suggests that GPT's responses are similar to the ground truth, regardless of the specific prompt scenario. In addition, GPT3.5 had more consistent naming for KCs compared to Llama2.

3 LIMITATIONS

When comparing GPT3.5 and Llama2, one significant difference is that GPT's system stores chat histories, making it easier to continue working from where the last session ended. In contrast, Llama2's system does not have this feature. This difference may explain why the KCs generated by Llama2 struggle with consistent KC naming and sometimes generate unrelated components. Additionally, Our study was limited to a single prompt per problem. We did not investigate potential variations in the outputs of the LLMs that might occur when multiple prompts are applied to the same problem.

4 DISCUSSION AND FUTURE WORK

Our initial findings suggest that LLMs effectively identify course-level KCs that closely match the expert labels. Our next step is to utilize the KCs gathered in this study to evaluate how effectively LLMs can identify KCs within a specific program and how the absence of a KC in a student program connects to misconceptions to provide immediate personalized feedback [2].

Additionally, future work will focus on improving the KC-extraction methodology that is presented in this work. This includes further prompt engineering and identifying the impact of including the potential list of course-level KCs in the prompt to achieve a more consistent and cohesive KC output from LLM. This enables instructors to get responses that use similar terminology as they present in their course when providing student feedback.

5 ACKNOWLEDGMENTS

This research was supported by the National Science Foundation (NSF) under Grants DUE-2236195 and DUE-2331965. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] Stephen H Edwards and Krishnan Panamalai Murali. 2017. CodeWorkout: short programming exercises with built-in data collection. In *ITiCSE'17*. 188–193.
[2] Muntasir Hoq, Jessica Vandenberg, Bradford Mott, James Lester, Narges Norouzi, and Bita Akram. 2024. Towards Attention-Based Automatic Misconception Identification in Introductory Programming Courses. In *SIGCSE'24 V. 2*.
[3] Kenneth R Koedinger, Albert T Corbett, and Charles Perfetti. 2012. The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive science* 36, 5 (2012), 757–798.
[4] Charles Koutcheme, Sami Sarsa, Juho Leinonen, Arto Hellas, and Paul Denny. 2023. Automated Program Repair Using Generative Models for Code Infilling. In *AIED'23*. 798–803.
[5] Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J Malan. 2024. Teaching CS50 with AI. (2024).
[6] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic generation of programming exercises and code explanations using large language models. In *SIGCSE'22 V. 1*. 27–43.
[7] Yang Shi, Robin Schmucker, Min Chi, Tiffany Barnes, and Thomas Price. 2023. KC-Finder: Automated Knowledge Component Discovery for Programming Problems. *International Educational Data Mining Society* (2023).
[8] J.D. Zamfirescu-Pereira, Laryn Qi, Bjoern Hartmann, John DeNero, and Narges Norouzi. 2023. Conversational Programming with LLM-Powered Interactive Support in an Introductory Computer Science Course. In *GAIED @NeurIPS'23*.

Data for Healthy Communities

A Public Interest Pilot Course Designed to Develop K-12 Data Literacy

Emily Nutwell
The Ohio State University
Columbus, Ohio, USA
nutwell.1@osu.edu

Kelsey Badger
The Ohio State University
Columbus, Ohio, USA
badger.60@osu.edu

Jessica T. Kulp
The Ohio State University
Columbus, Ohio, USA
kulp.95@buckeyemail.osu.edu

ABSTRACT

Data analytics skills are in high demand in a wide variety of professions and data is increasingly present in our everyday lives. Considering this, educators should be equipped to prepare future data scientists, but also data literate citizens. This project engages high school students in the real-world practice of data analytics to promote the public interest. The research team is partnering with a local public STEM high school to design, evaluate, and publicly share an enrichment course that introduces students to data analytics in the context of public health. Integrating data science education and public health will enable students to engage in meaningful data-driven problem solving while promoting civic engagement. A key challenge in introducing data skills in K-12 is addressing barriers to access. To cope with these barriers, the course design incorporates the use of spreadsheets to engage students in computational thinking. This poster will present preliminary assessment results from the 3-week enrichment course conducted in January 2024. This project contributes to the ongoing discussion of K-12 data science education by demonstrating the value of contextualizing data science as a key competency for 21st century public service.

ACM Reference format:

Emily Nutwell, Kelsey Badger, and Jessica T. Kulp. 2024. Data for Healthy Communities: A Public Interest Pilot Course Designed to Develop K-12 Data Literacy. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages.
https://doi.org/10.1145/3626253.3635589

1 INTRODUCTION & BACKGROUND

As data becomes more prevalent in all aspects of society, introducing students to data literacy concepts in the high school years is becoming increasingly important. Data literacy can be understood as the ability to work with data through an inquiry process to address real-world issues, including knowledge and skills related to collection, preparation, analysis, visualization, communication, and critique of data and results [11]. Educators need to prepare students to be proficient in applying these skills across all aspects of their personal and professional lives, including as engaged citizens [4, 10].

Preparing a data literate citizenry also ensures a workforce that can apply data-driven problem solving to social and cultural problems, which are complex and typically involve several interconnected issues and constraints. Since data is present in every field, it is imperative that students looking to choose careers or majors for college understand how data is integrated in all professions, including public service roles that may not be historically associated with data or computing. Public interest technology is an emerging field of interdisciplinary technologists focused on applying data and technology in the service of social good [6]. The development of a workforce adept at understanding social problem contexts and data-driven approaches is increasingly urgent as the world grapples with global health threats such as pandemics, environmental hazards, the mental health crisis, and antibiotic resistance [5].

However, there are significant barriers preventing equitable access to data education across the country. Many of these relate to an ongoing debate about when and how data skills should be introduced in the high school curriculum. Due to the use of computational tools like Python and R in many popular data science curricula, exposure to data education is often contingent on the availability of computer science courses. Ohio ranks near the middle in the country in computer science course availability, with 48% of public high schools offering a class in computer science [2]. Yet only 3.6% of Ohio high school students took a computer science class during the 2021-2022 school year [3]. To improve access, it is necessary to both increase computer science course availability and to embed foundational data concepts across the curriculum.

Open educational resources (OERs) provide teachers a streamlined approach to experimenting with new material in existing courses. However, there is a dearth of OERs that do not rely on the use of advanced computational tools to introduce data concepts. One way to lower this barrier is to develop data science modules using spreadsheets, a more universal tool that can be used to teach fundamental computational concepts such as basic sequences, functions, and variables [7]. Spreadsheets offer a compelling opportunity to introduce data science concepts across the curriculum while eliminating a steep learning curve required for more specialized tools.

2 COURSE OVERVIEW

The Data for Healthy Communities (DHC) course is a pilot project of spreadsheet-based modules that combines the need for

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635589

more accessible data science curricula with the values of public interest technology. We aim to broaden student horizons regarding the use of data for social good and to create open educational resources that introduce data science using approachable, spreadsheet-based methods.

DHC is a three-week enrichment course with approximately 15 instructional hours. It was developed in close collaboration with a diverse STEM high school whose student body is drawn from over two dozen Central Ohio school districts. It will be taught in January 2024 during a period where students are offered the opportunity to explore extra-curricular interests through an array of short elective offerings. The course will be evaluated using a pre/post-test that incorporates both cognitive assessment and attitudinal measures of student interest and perceived self-efficacy towards data science.

The course is designed to introduce high school students to data science concepts in the context of public health. These topics, such as high infant mortality, poor air quality, and the opioid epidemic are all prevalent problems in today's society. We hope to engage these students in understanding the connections between their own communities and technology in the public interest, helping them envision educational paths and career opportunities where they are empowered to address long-standing inequities through data-driven problem solving.

A backwards course design process was used to develop a series of modules, each of which introduces a skill tied to one or more learning objectives [9]. These objectives were created in part through evaluation of public health workforce development programs and in collaboration with a high school teacher. They are also mapped to one or more of the Ohio Department of Education's standards, including those for Data Science Foundations, Statistics and Probability, and Algebra. This breadth of mappings is intended to facilitate future transfer of the curriculum to use during the regular school year.

The first modules of the course have students explore public interest technology careers and consider how data can be used to explore health inequities in their communities. The remainder of the course modules guide students through the process of the data lifecycle using various sources of air quality data [1]. Students will collect readings using sensors and organize their data according to tidy data principles [8]. They will then be introduced to more complex, real-world data from the Canadian wildfires during the summer of 2023. Students will practice using spreadsheet functions and features for selecting, summarizing, and visualizing data.

Collectively, the skill-building modules demonstrate how an argument can be supported with data. This provides students an example to apply in the completion of a course project. The project features a fictional case-study about the use of county funds to improve public health through investment in a community garden. Students will be asked to recommend which neighborhood should be prioritized for the garden, using a real-world dataset of local indicators as evidence. To highlight the importance of effective visualizations, they will be asked to support their argument with a chart or graph. Following group presentations, the instructors will guide students through the implications of using data to persuade, rather than as definitive proof of a single correct response.

3 CONTRIBUTIONS & FUTURE WORK

This project fills an important gap in data science open educational resources (OERs) that are taught using accessible spreadsheet-based methods. Using this approach, teachers across the curriculum can engage students with data literacy concepts and basic computing skills outside of computer science courses.

It is also unique in its emphasis on problem-solving using open government data. Experience with messy real-world data develops needed familiarity with the challenges of authentic problem-solving. However, a key challenge is curating datasets such that they are accessible for secondary education while still retaining the complexity of the real-world. To expand this educational model, further work is needed to curate data appropriate to various public interest technology contexts. The hope is that the framework presented here will contribute to the development of a citizenry and workforce that is prepared to use data to solve complex societal challenges.

ACKNOWLEDGMENTS

This work is based upon work supported by New Venture Fund under NVF-PITU-The Ohio State Univers-Subgrant-018811-2022-10-01. We also gratefully acknowledge our colleagues Chris Orban and Wonjung Hwang who collaborated on course design.

REFERENCES

[1] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rüdiger Wirth. 1999. *CRISP-DM 1.0.* SPSS Inc.

[2] Code.org. 2023. *Support K-12 Computer Science Education in Ohio.* Promote Computer Science.

[3] Mike Duffey, John Wiseman, and Kelly Evans Gaier. 2022. *Report of the State Committee on Computer Science.* Ohio Department of Education.

[4] Engida Gebre. 2022. Conceptions and perspectives of data literacy in secondary education. *British Journal of Educational Technology* 53, 5 (2022), 1080–1095. DOI:https://doi.org/10.1111/bjet.13246.

[5] Olga Jonas, Ian Parry, Dan Chrisholm, Nick Banatvala, and Ramanan Laxminarayan. 2014. Global health threats of the 21st century. *Finance & Development* 51, 4 (December 2014). International Monetary Fund.

[6] Tara Dawson McGuinness and Hana Schank. 2021. *Power to the public: The promise of public interest technology.* Princeton University Press, Princeton, NJ.

[7] Marcos Román-González, Juan-Carlos Pérez-González, and Carmen Jiménez-Fernández. 2017. Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior* 72 (2017), 678–691. DOI:https://doi.org/10.1016/j.chb.2016.08.047.

[8] Hadley Wickham. 2014. Tidy data. *Journal of Statistical Software* 59, 10 (Sep. 2014), 1–23. DOI:https://doi.org/10.18637/jss.v059.i10.

[9] Grant Wiggins and Jay McTighe. 1998. *Understanding by design.* Association for Supervision and Curriculum Development, Alexandria, VA.

[10] Alyssa Friend Wise. 2020. Educating data scientists and data literate citizens for a new generation of data. *Journal of the Learning Sciences* 29, 1 (2020), 165–181. DOI:https://doi.org/10.1080/10508406.2019.1705678.

[11] Annika Wolff, Daniel Gooch, Jose J. Cavero Montaner, Umar Rashid, and Gerd Kortuem. 2016. Creating understanding of data literacy for a data-driven society. *The Journal of Community Informatics* 12, 3 (2016), 9–26. DOI:https://doi.org/10.15353/joci.v12i3.3275.

Python Programming Education with Semantics-oriented Screen Reading for K-12 Students with Vision Impairments

God'salvation F. Oguibe
The University of Texas at San Antonio
USA
Godsalvation.Oguibe@my.utsa.edu

Lauryn M. Castro
The University of Texas at San Antonio
USA
Lauryn.Castro@my.utsa.edu

Katherine Cantrell
The University of Texas at San Antonio
USA
Katherine.Cantrell@utsa.edu

Kathy B. Ewoldt
The University of Texas at San Antonio
USA
Kathy.Ewoldt@utsa.edu

Leslie Cockerill Neely
The University of Texas at San Antonio
USA
Leslie.Neely@utsa.edu

Wei Wang
The University of Texas at San Antonio
USA
Wei.Wang@utsa.edu

ABSTRACT

Because of the high pay and high flexibility, computer science careers can be highly viable for people with blindness or vision impairments (BVI). However, in our programming education, we observed that existing screen readers used by students with BVI usually cannot properly handle computer programs, which mix English letters, digits, and punctuation marks. When applied to computer programs, current screen readers either ignore the punctuation marks, or mix English words, digits, and punctuation marks, making the screen reading either incorrect or hard to understand. The resulting difficulty in understanding program statements significantly hinders students' ability to locate incorrect code and independent coding.

To address these limitations in current screen reading, we are developing a new semantic-oriented screen reader, called *JupyterVox*, which reads Python statements by their meanings to speedup code navigation, and thus improve independent coding skills. This new screen reader is based on compiler's lexical and syntax analyses to parse, understand, and read program statements. This poster presents the initial implementation and results of *JupyterVox*.

CCS CONCEPTS

• **Human-centered computing** → **Accessibility technologies**; • **Social and professional topics** → **Software engineering education**; • **Software and its engineering** → **Compilers**.

KEYWORDS

Special Education, Computer Programming Education, Blindness and Vision Impairment

ACM Reference Format:
God'salvation F. Oguibe, Lauryn M. Castro, Katherine Cantrell, Kathy B. Ewoldt, Leslie Cockerill Neely, and Wei Wang. 2024. Python Programming Education with Semantics-oriented Screen Reading for K-12 Students with Vision Impairments. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635494

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635494

1 INTRODUCTION

Software engineers are among the highest-paid occupations in the US. In addition to the high pay, software engineering jobs are also flexible in working modality. A significant number of software engineering positions are remote and only require a desk, a computer, and the Internet. These combined benefits make software engineering a perfect position for people with disabilities, including people with blindness or vision impairments.

In the past years, the authors have been teaching simple Python programming and machine learning to middle and high school students with BVI. A major difficulty we and our students have encountered in our teaching is the confusing readings from existing screen readers. Screen readers are typically designed to read English paragraphs instead of computer program statements, which mix letters, digits, and punctuation marks. The computer statement readings from current screen readers are typically difficult to understand – the readings can be either complex and/or misleading. A direct result of this understanding difficulty is the slow code reading and code navigation, which directly leads to slow programming speed and low independent coding abilities. Some students would completely avoid using screen readers when coding and chose to use the more expensive Braille displays. More specifically, there are at least two issues with current screen readers,

(1) First, by default, most screen readers will ignore (some of) the punctuation marks when reading. For example, as shown in Figure 1, for the dictionary definition statement {"$t1$" : a, "$t2$" : b}, Windows Narratoronly reads "T one A two twelve." Mac VoiceOverreads braces, but not quote marks. These readings completely misrepresented the statement.
(2) Second, when the screen readers are configured to read punctuation marks, the readings become long and confusing. Figure 1 also shows the screen readings of Windows Narrator and Mac VoiceOver when they are configured to read all

	Standard Screen Reader			*JupyterVox*		
Type of Statement	Statement	Time to Understand	Understanding Correctness	Statement	Time To Understand	Understanding Correctness
Expression	a*b+c	10.24s	correct	a*(b+c)	9.67s	correct
Assignment	z=y%3//5	21.71s	incorrect	b=a//2 % 3//5	11.84s	correct
Aug Assignment	x +=5	6.82s	correct	b -= a	7.48s	incorrect
Assign w. Expr.	c=3-x*y	34.42s	incorrect	c = y/z+3	6.80s	correct
Complex Assign.	a=(x*y)+(z/5-b)	10.37s	incorrect	c = (a*b) - (3*z/y)	43.7s	correct
if statement	if x<=10: return 9	7.12s	correct	if x>=3: return 4	8.87s	correct
String Variable	test_string="Hello World"	7.19s	incorrect	day_string="Sunday"	10.80s	correct

Table 1: Initial evaluation results.

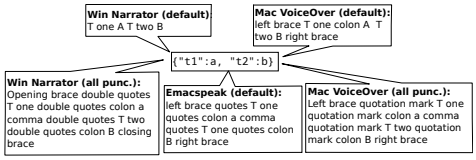


Figure 1: An example of how the handling of punctuation marks affect screen readings for three screen readers under the default and read-all-puncutations (all punc.) settings.

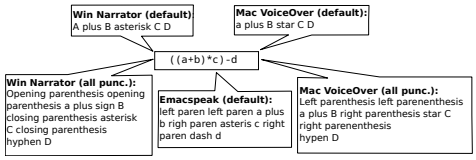


Figure 2: Second example of how the handling of punctuation marks affect screen readings, especially for statements with nested parentheses.

punctuation marks. Additionally, Figure 1 includes the reading of Emacspeak [4], which is a screen reader that understands programs and reads punctuation marks by default. Figure 1 shows that all these readings mixed punctuation marks, digits, and English words, and require considerable effort to parse in one's mind. This problem is even more complicated when nested parentheses are involved (e.g., Figure 2) – determining the matching pairs of parentheses while they are read is extremely difficult for beginners.

Two other popular screen readers, Jaws [2] and NVDA [1], also have the above limitations. There is also work on using audio-assisted debugging [6, 7] and more intuitive languages [3, 5, 8]. However, these studies cannot be directly applied to Python, which is extensively used in Data Science.

2 METHODOLOGY

To address these limitations, our hypothesis is that: *Instead of reading literally, the understanding of program statements can be significantly simplified if the statements can be read based on their meaning (semantics), which, in turn, accelerates code navigation and improves independent coding skills for programming education.*

As lexical and syntax analyses in compilers already understand the semantics of program statements, it is then possible to employ compiler techniques to generate semantics-oriented readings. More specifically, we employ Python AST module to generate an abstract syntax tree (AST) given a statement, then convert that AST tree to an English sentence representing the meaning of that statement. This conversion closely resembles the coding generation processing

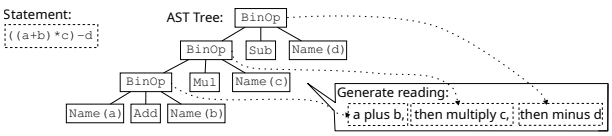


Figure 3: An example of reading generation using AST tree in *JupyterVox* using the same statement from Figure 2.

using an AST tree. Figure 3 gives an example of how speeches are generated based on an AST tree for the same statement in Figure 2. The reading is "a plus b, then multiply c, then minus d."

3 EVALUATION

We finished an initial implementation of the screen reader, called *JupyterVox*. As an early evaluation of *JupyterVox*, we conducted a short experiment with one human subject who listened to seven statements read by the an existing screen reader and *JupyterVox*. Table 1 gives the results of this screen reader. The subject was considered as having understood the reading once he could restate the statement correctly. As Table 1 shows, *JupyterVox* could significantly increase the correctness of statement understanding, from only three statements to six statements.

ACKNOWLEDGEMENT

This work is supported by NSF grants, 2202632, 2221843, 2155096, and 2215359. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied of NSF.

REFERENCES

[1] NV Access. 2023. About NVDA. https://www.nvaccess.org/about-nvda/. (2023).
[2] Freedom Scientific Inc. 2023. JAWS. https://www.freedomscientific.com/products/software/jaws/. (2023).
[3] Aboubakar Mountapmbeme and Stephanie Ludi. 2020. Investigating Challenges Faced by Learners with Visual Impairments Using Block-Based Programming/Hybrid Environments. In *Int'l ACM SIGACCESS Conference on Computers and Accessibility.*
[4] T.V. Raman. 1997. Emacspeak-an Audio Desktop. In *IEEE COMPCON.*
[5] Jaime Sánchez and Fernando Aguayo. 2006. APL: Audio Programming Language for Blind Learners". In *Computers Helping People with Special Needs*, Klaus Miesenberger, Joachim Klaus, Wolfgang L. Zagler, and Arthur I. Karshmer (Eds.).
[6] Andreas Stefik, Roger Alexander, Robert Patterson, and Jonathan Brown. 2007. WAD: A Feasibility study using the Wicked Audio Debugger. In *IEEE Int'l Conf. on Program Comprehension.*
[7] Andreas Stefik, Andrew Haywood, Shahzada Mansoor, Brock Dunda, and Daniel Garcia. 2009. SODBeans. In *IEEE Int'l Conf. on Program Comprehension.*
[8] Andreas M. Stefik, Christopher Hundhausen, and Derrick Smith. 2011. On the Design of an Educational Infrastructure for the Blind and Visually Impaired in Computer Science. In *ACM Technical Symp. on Computer Science Education.*

Motivating and Increasing the Relevance of Elementary Computer Science in Rural Communities

Exploring Elementary Teacher Perspectives in Idaho

Diana Dajung Oh
Kathryn M. Rich
American Institutes for Research
Chicago, IL USA
{doh, krich}@air.org

Kathleen L. Bosch-Wilson
Idaho STEM Action Center
Boise, ID USA
katie.boschwilson@stem.idaho.gov

Heather L. Cunningham
BootUp Professional Development
Austin, TX USA
heather@bootuppd.org

ABSTRACT

The Idaho Elementary Computer Science (CS) Collaborative (IECC) is a researcher-practitioner partnership dedicated to understanding how to better support the predominately rural districts in Idaho to implement elementary CS education. As part of the IECC's work, we conducted a survey of elementary teachers across the state. This poster will present our initial findings. Two of the research questions we addressed via the survey are: (1) What factors would motivate Idaho elementary teachers to pursue professional development (PD) related to CS? (2) What are elementary teachers' perceptions of the value of connecting CS education to rural community economies and industries? Descriptive analyses of 309 survey responses suggest that about three-quarters of elementary teachers reported that the following factors would be at least somewhat influential in their decisions to pursue CS PD: a K-5 computational thinking requirement, a district-wide CS planning staff, a pathway for elementary teachers to earn a certification in CS teaching, and a way to request CS PD funding. Few (9%) Idaho elementary teachers found that the perception of a disconnect between CS education and rural community interests was a significant barrier to elementary CS education. Even so, over 70% of them were moderately to strongly interested in mechanisms for connecting their CS education efforts to rural community industries through guest speakers and industry partnerships. The poster will discuss the implications of these and other emerging findings from the teacher survey, as well as IECC's related surveys of Idaho district superintendents and elementary school principals.

ACM Reference format:

Diana Dajung Oh, Kathryn M. Rich, Kathleen L. Bosch-Wilson, & Heather L. Cunningham. 2024. Motivating and Increasing the Relevance of Elementary Computer Science in Rural Communities: Exploring Elementary Teacher Perspectives in Idaho. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635500

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03. https://doi.org/10.1145/3626253.3635500

1 BACKGROUND

The Idaho Elementary CS Collaborative (IECC) is a researcher-practitioner partnership among the Idaho STEM Action Center (Idaho STEM AC), BootUp Professional Development (BootUp), and the American Institutes for Research (AIR) dedicated to investigating problems of practice and effective supports for developing elementary CS education pathways in Idaho. More than 100 of Idaho's 116 school districts are in rural areas. Ergo, much of the IECC's work focuses on addressing challenges and leveraging strengths of rural communities when developing elementary CS education efforts. A key activity in the first year of the IECC was to conduct a survey of elementary teachers in Idaho to gather information about existing elementary CS instruction and perceptions of barriers and needed supports. We present preliminary analyses of the survey data in this poster.

2 RATIONALE FOR RESEARCH QUESTIONS

Rural communities (as defined by the National Center of Education Statistics locale definitions and in [4]), including many Idaho communities, often have specific needs related to CS education. For example, adequate staffing for CS and other STEM education efforts is a challenge for rural communities. There is a shortage of STEM teachers nationwide, and the need is particularly acute in rural areas [1]. This situation often leads to teachers with limited background in STEM and STEM education teaching STEM, which can place additional stress on teachers and lead to greater teacher turnover [4]. This also illustrates the importance of providing CS PD to teachers in rural communities. Ergo, we ask the research question, what factors would motivate Idaho elementary teachers to pursue PD related to CS?

Students in rural communities may also have relatively low exposure to CS careers and how those careers can benefit their communities. STEM education is not often perceived as connected to rural towns, geographic areas, or industries. Rather, interest in and success with STEM education is associated with students leaving their communities after high school [3, 4]. This perception may extend to CS. Even when CS courses are available to students in rural communities, enrollment in the courses is not as predictive of student interest in CS as it is for students in urban areas [2]. As such, we ask the second research question, what are elementary

teachers' perceptions of the value of connecting CS education to rural community economies and industries?

3 SAMPLE AND METHOD

The IECC distributed the survey via Idaho STEM AC's monthly newsletters and e-mails as well as at in-person educator PD events such as i-STEM, Idaho Education Technology Association Conference, and Idaho Association of School Administrators Conference. Respondents were 309 teachers (92% certified teachers, 8% classified staff). The majority were white (89%). Teachers on average had 11.28 teaching years (*range* = 0-35, *median* = 10), and 4.43 years teaching CS (*range* = 0-29, *median* = 1). Teachers taught in districts across the state of Idaho, with representation across all six of Idaho's regions.

To explore what would motivate teachers to seek out CS PD, teachers were asked "to what extent would each of the following influence your interest in seeking PD related to CS?" and were presented the six factors in Table 1 to which they responded with a 4-point Likert scale (*1 = not at all, 4 = very much*).

To understand teachers' perceptions about connecting CS education to rural community economies and industries, teachers were asked the following three questions: 1) how much is the perception that CS education is connected to students leaving rural communities a barrier to developing and/or implementing plans for CS instruction in your classroom? 2) How much would you be interested in opportunities to host guest speakers from local industries to talk to students about how they use CS? and 3) how much would you be interested in opportunities to learn more about how to connect CS education to your local community? Teachers used 4-point Likert scales (i.e., *1 = not a barrier, 4 = a significant barrier*; *1 = not at all interested, 4 = very interested*) to answer the questions. Descriptive analyses focusing on the spread and distribution of teachers' responses were conducted.

4 RESULTS

Table 1 shows that teachers rated "a way to request funding specifically for CS PD" as the most influential factor on their motivation to pursue CS PD, with 45% rating it as very motivating and 33% as somewhat motivating.

Table 1: factors that would motivate teachers to seek CS PD

Item description	Mean	SD
A CS requirement for high school graduation	2.25	1.03
A state assessment of CS	2.31	1.05
A K-5 computational thinking and digital literacy requirement	3.08	.86
A staff member with time dedicated to planning and supporting district-wide CS instruction	3.12	.83
A pathway for elementary teachers to earn a certification in CS teaching	3.09	.95
A way to request funding specifically for CS PD	3.17	.91

Other influential factors in Table 1 include a CS-related K-5 requirement, a staff member to support CS, and a pathway for elementary teachers to earn a certification in CS teaching, with nearly three quarters of the teachers rating these factors as very (36-43%) or somewhat (31-40%) motivating. Fewer teachers rated a high school CS requirement (12%) and a state assessment of CS (15%) as very motivating with about a third of them saying they're not at all motivating (30-32%). This may be in part because they are less relevant to elementary school teachers. The results suggest that K-12 CS pathways are not likely to reach K-5 spaces if only a top-down approach is taken to motivate elementary teachers.

For the connection between CS education and rural communities, Figure 1 shows most teachers (72%) did not view "perceptions that CS education is connected to students leaving rural communities" as a barrier or significant barrier to elementary CS education—contrary to our expectations based on other rural STEM education research [3, 4].

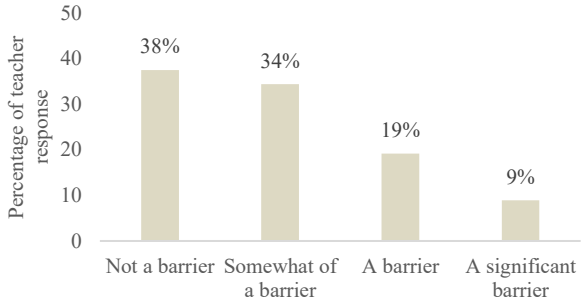


Figure 1: Teacher responses to the extent to which "perception that CS education is connected to students leaving rural communities" is a barrier to elementary CS education.

Teachers did not perceive CS education connected to students leaving their rural communities, but rather expressed interest in making connections between CS and the local community. Most teachers were moderately (32-34%) or very (39-40%) interested in "opportunities to host guest speakers from local industries to talk to students about how they use CS" and "opportunities to learn more about how to connect CS education to my local community." This result suggests the importance of place-based CS education in K-5 for rural communities in Idaho.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation #2219333.

REFERENCES

[1] Devon Brenner, Amy Price Azano, and Jayne Downey, 2021. Helping new teachers stay and thrive in rural schools. *Phi Delta Kappan* 103, 4, 14-18. DOI: https://doi.org/10.1177/00317217211065821

[2] Gallup. 2021. *Developing careers of the future: A study of student access to, and interest in, computer science.* https://www.gallup.com/analytics/354740/ study-of-student-interest-in-computer-science.aspx

[3] Joni M. Lakin, Tamra Stambaugh, Lori M. Ihrig, Duhita Mahatmya, and Susan G. Assouline, 2022. Nurturing STEM talent in a rural setting. *Phi Delta Kappan* 103, 4, 23-30. DOI: https://doi.org/10.1177/00317217211065823

[4] Daniel Showalter, Robert Klein, Jerry Johnson, and Sara L. Hardman,2017. *Why rural matters 2015-2016: Understanding the changing landscape.* Rural School and Community Trust. https://files.eric.ed.gov/fulltext/ED590169.pdf

Web-based Course Assessment System of Student Learning Outcomes: A Pilot Study

Daniel Ojeda
Department of Computer Science and Technology
Kean University
Union, NJ
ojedada@kean.edu

Patricia Morreale
Department of Computer Science and Technology
Kean University
Union, NJ
pmorreal@kean.edu

Daehan Kwak
Department of Computer Science and Technology
Kean University
Union, NJ
dkwak@kean.edu

ABSTRACT

The assessment of student learning is essential, with instructors using end-of-course assessments against Course Objectives and Student Learning Objectives (SLOs), typically through assignments and exams against a predetermined benchmark. However, the reliance on outdated paper-based systems can delay crucial information access and lead to errors and inconsistencies in assessments. Therefore, a pilot study is underway to explore the benefits of transitioning from paper-based to electronic-based assessment systems. This shift aims to facilitate easier monitoring and more consistent tracking of course objectives, potentially enhancing the quality of courses and curriculum. All faculty-submitted assessment data is securely stored on a server, which supports the creation of an interactive data visualization tool. This tool is primarily designed to aid faculty in making informed decisions to better engage, motivate, and improve the learning experience for students. Moreover, the electronic system promises significant efficiency improvements by streamlining information retrieval and standardizing annual assessment processes.

CCS CONCEPTS

• Social and professional topics → Professional topics → Computing education → Student assessment.

KEYWORDS

Course assessment system, computer science education, student learning objectives, ABET

ACM Reference format:
Daniel Ojeda, Patricia Morreale and Daehan Kwak. 2024. Web-based Course Assessment System of Student Learning Outcomes: A Pilot Study. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635587

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635587

1 INTRODUCTION

The motivation for developing the Faculty Course Assessment system is to assist faculty in fine-tuning students' performance using an interactive data visualization tool. However, the most crucial aspect of the assessment system is its capability to address the challenge of inconsistent faculty course assessment submissions, providing a solution for easy access to submit and maintain information related to Assessment Materials.

2 RELATED WORK

Over the years, several course assessment systems have been developed to optimize learning performance, addressing different aspects of the course assessment process [1-4]. However, none of these systems currently incorporate real-time interactive data visualization tools. Our system sets itself apart from others by offering a unique combination of features that makes the Faculty Course Assessment system a highly valuable addition to the domain of course assessment. Its built-in automated error-proof mechanisms and the ability to perform real-time semester-over-semester comparisons make it a robust faculty support tool. Furthermore, the inclusion of tutorial videos demonstrates a friendly user interface and aiding user adaptation, ensuring that faculty members of all technological proficiency levels can easily adapt to and effectively utilize our system in real-time.

3 COURSE ASSESSMENT SYSTEM

The research team designed a course assessment system for the Computer Science department to enhance and optimize feedback on learning performance. This system, accessible through a URL depicted in Figure 1, not only stores course information but also prioritizes ease-of-use and supports outcomes-based analysis for continuous improvement, curriculum mapping, and ABET accreditation. Faculty members, after accessing the system, are presented with two main options: the Faculty Course Assessment Report for submitting assessment materials, as shown in Figure 2, and Visualization for displaying student performance against SLOs, illustrated in Figure 3. Faculty receive an email confirmation upon submission of assessment materials and graded student work samples, and they can select specific courses or multiple sections for comparison, as well as start and end terms for the visualization of SLO achievement percentages.

Based on feedback over three semesters, several improvements were implemented: 1) Implementing a standardized format for assessment score entry to address inconsistencies caused by varied instructor input methods. 2) Incorporating visual cues and confirmation messages to correct frequent errors in section number selection, enhancing data accuracy. 3) Adding an 'information' button and instructional videos to clarify student count issues and participation criteria for SLO assessments, reducing confusion about which students to include in assessments. 4) The system now automatically detects and displays the current semester, addressing the problem of instructors selecting the wrong year or semester. 5) Promoting the practice of comparing current assessment scores with previous years to foster continuous improvement. 6) Acknowledging the varied learning curve among faculty members in adapting to web-based systems. 7) Clarifying the distinction between course learning outcomes and ABET SLOs with tutorial videos, ensuring faculty understand and apply these concepts correctly. 8) These enhancements aim to streamline the assessment process, improve data reliability, and facilitate a more intuitive user experience for faculty, thereby supporting the department's goals for curriculum development and accreditation.

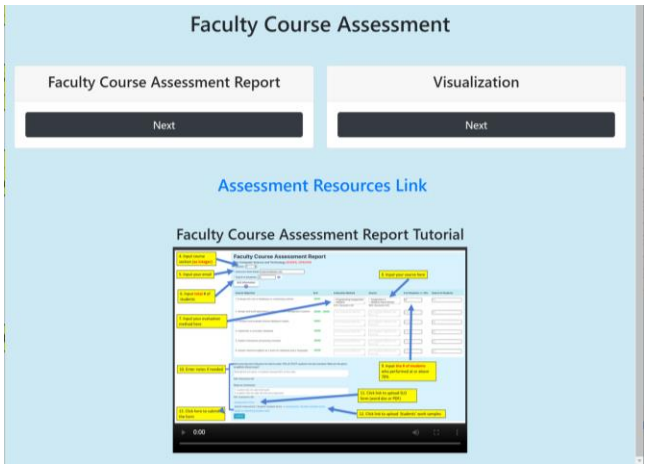


Figure 1: Course Assessment System Home Page.

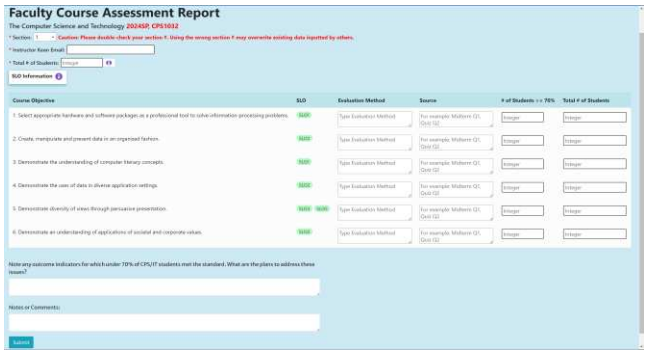


Figure 2: Faculty Course Assessment Report.

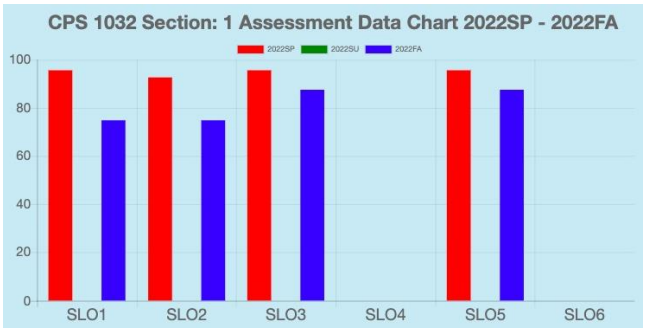


Figure 3: Student Learning Outcome Visualization.

4 CONTRIBUTIONS AND FUTURE WORK

The Computer Science department benefits from an automated course assessment system, enhancing instruction quality monitoring. This system provides faculty with current, accurate, and detailed visual reports on courses, allowing for performance analysis and identification of potential learning effectiveness issues. The transition from paper-based systems to an online tool for consistent student work collection and student learning outcomes input, aimed at ABET accreditation, received positive feedback from the department. Despite varying learning curves among instructors using the online system, a comfortable proficiency is attained after three rounds of assessment submissions. The system's visualization capabilities, automated data checks, and comparative analyses semester-over-semester significantly endorse its adoption. Future enhancements may extend its application to other departments, leveraging its established benefits. This iterative development process promises ongoing improvements to further streamline faculty workflow and enhance educational outcomes.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1834620 and 1928452.

REFERENCES

[1] T Tonghui Xu, Hsien-Yuan Hsu, Melissa Nemon, Christopher Hansen, John Hunter Mack, and David J. Willis. 2023. A tag-based framework for collecting, processing, and visualizing student learning outcomes. *ASEE Annual Conference & Exposition*, Baltimore, Maryland. https://peer.asee.org/42533

[2] Christopher Nitta and Kurt Eiselt. 2021. Using a Comprehensive Third-Party Exam for ABET Student Outcome Assessment. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (SIGCSE '21). ACM, New York, NY, USA, 673–678. https://doi.org/10.1145/3408877.3432529

[3] Ken Christensen, Rafael Perez, Purushottam Panta, and Pummy Bedarahally. 2011. Unifying program-level ABET assessment data collection, analysis, and presentation. *2011 Frontiers in Education Conference (FIE)*. https://doi.org/10.1109/fie.2011.6142799

[4] Aamir Shafi, Saqib Saeed, Yasser A. Bamarouf, Sardar Zafar Iqbal, Nasro Min-Allah, and Mohammed A. Alqahtani. 2019. Student Outcomes Assessment Methodology for ABET Accreditation: A Case Study of Computer Science and Computer Information Systems Programs. *IEEE Access*, 7, 13653–13667. https://doi.org/10.1109/access.2019.2894066

Blurring the Boundaries between Science, Math and Computer Science

Chris Orban
orban@physics.osu.edu
Ohio State University
Columbus, Ohio, USA

ABSTRACT

In 2017 the STEMcoding project was launched, which aimed to develop and disseminate free content that mixes content from science, math and computer science standards. Early work focused on "physics of video games" activities for high school physical science and physics while later work has ventured into content for astronomy and high school mathematics. Some of these activities appear on hourofcode.com and in "instructional supports" for physical science in Ohio. An important component of our efforts is creating high quality videos that walk through our activities on the STEMcoding youtube channel (http://youtube.com/STEMcoding). These videos frequently feature women and students from underrepresented groups. Ultimately our goal is to encourage high school science and math teachers to integrate computer science and data science content into their subjects, even if they do not have the credentials to teach computer science. This can enhance science and math classes, and as these teachers become comfortable with this approach, some go on to become certified to teach computer science.

CCS CONCEPTS

• **Social and professional topics** → **Computational thinking**; **Model curricula**; **K-12 education**.

KEYWORDS

integration, curriculum, math, science

ACM Reference Format:

Chris Orban. 2024. Blurring the Boundaries between Science, Math and Computer Science. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635580

1 INTRODUCTION

In discussions of increasing the numbers and diversity of students who gain computer science skills, the focus is often on how many high school computer science teachers we have and how we can train more of them. While this is an important topic, computer science teachers are still far outnumbered by teachers of other subjects such as science and mathematics which are both highly technical subjects. In this poster, I will outline efforts to encourage high school science and math teachers to integrate computer science and data science into their subjects in a way that strongly aligns with existing learning goals.

There are many groups that work at the intersection of computer science and what is sometimes called subject integration. An incomplete list includes csandmath.org [4], Bootstrap [7], vpython [2], Let's Code Physics [3] and others. Perhaps the most significant effort to intersect computer science and math instruction at the high school level is introductory data science such as the UCLA introduction to data science curriculum [5] and Stanford's youcubed data science [10]. Also notable is the San Diego State University high school discrete math project [8]. As impressive as this list is, it is still dwarfed by the number of organizations that work to advance traditional computer science education.

In this poster, we outline efforts in our organization to "blur the lines" between math, and science and computer science content. Typically this involves simple, javascript-based coding tasks using the graphics library p5.js which is designed to facilitate even absolute beginner programmers. Some of the activities we have designed involves both a javascript component and a spreadsheet component, and some activities only involve spreadsheets.

In terms of design philosophy, we are influenced by the Next Generation Science Standards (NGSS) [9] which places computational thinking on essentially the same level as mathematical reasoning as being foundations that science content can build from. There are many papers in the literature that describe or attempt to define computational thinking. We are strongly influenced by Weintrop et al[11] and Orban & Teeling-Smith[6]. The Weintrop et al.[11] describes "Data practices" as one of the key elements of computational thinking. So we do not view data science as being at odds or in tension with computational thinking in any way.

2 RESOURCES

Since we have been building activities since 2017 and even earlier, it is difficult to briefly summarize them all. The activities we produce all have interesting visual components and some have interesting interactivity, such as the physics of video games activities mentioned earlier for physical science and physics. A key reason we use javascript with p5.js is because it does a great job of rendering crisp 2D graphics with a high frame rate even on old chromebook machines. A math relevant example with interesting visual components is our "Pi day" activity which uses a monte carlo approach to estimate the digits of Pi which involves using a random number generator (https://hourofcode.com/stempiday). In the Pi day activity, students will view the random positions generated by

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635580

the random number generator as dots on the screen and the dots inscribed by the circle are given a different shade than the ones outside of it, so that an image naturally emerges as the program better estimates the digits of pi.

Another recurring theme in our activities is what in computational science is referred to as “verification" tests. In a computer science context this would be referred to as “unit tests". Many of our activities include both an interesting visual or interactive component and a verification test. An example is our Pong game (https://hourofcode.com/stempong) where students check to see if the speed is the same before and after the collision with the wall (as energy conservation demands). Another example is an activity we designed after the classic Lunar Lander game (https://hourofcode.com/stemplanetoids). Students create a playable game but they also take time to quantitatively check if the rocket falls like a projectile would.

Other activities are variations on these themes. For example, there are activities where we use a random number generator in javascript with p5.js to generate both an interesting visual and a CSV file. The student can then analyze the CSV data in a spreadsheet. Our activities for Earth Day and another astronomy activity follow this pattern.

3 PROFESSIONAL DEVELOPMENT

We have organized teacher professional development in our state and nationally in partnership with the American Association of Physics Teachers since 2016 [1]. During summer 2020 and 2021, there was tremendous interest in high quality, standards aligned activities that can be completed completely online. An interesting observation from these experiences is that some teachers receive training and then wait a full year before implementing any activities that connect computer science and data science to their subjects. We have also noticed that the confidence building and skills that teachers gain during our workshops helps them to assist with after school coding programs (e.g. girls who code) or lead them to pursue becoming fully credentialed to teach computer science in addition to math or science.

4 VIDEO CONTENT

We have also worked hard to create video content that walks through our activities. These videos help students and teachers alike and they help to overcome the challenge of covering both science/math and computer science related content at essentially the same time. Importantly these videos, which are available on our youtube channel (http://youtube.com/STEMcoding), include women and students from underrepresented groups. An important strategy in this work is partnering with student organizations like the local chapter of the National Society of Black Engineers and the Society for Women in Physics.

REFERENCES

[1] 2023. K12 Programs Online Courses for Grad Credit and CEUs. https://aapt.org/K12/Online_Courses_for_Grad_Credit_and_CEUs.cfm.

[2] Robert Beichner, Ruth Chabay, and Bruce Sherwood. 2010. Labs for the Matter & Interactions curriculum. *American Journal of Physics* 78, 5 (2010), 456–460. https://doi.org/10.1119/1.3266163

[3] Brian Lane. 2023. Let’s code physics youtube channel. https://www.youtube.com/@LetsCodePhysics.

[4] Mike Larson and Ashley Goetz. 2023. CS and Math. http://csandmath.org.

[5] University of California at Los Angeles. 2023. Introduction to Data Science. https://www.introdatascience.org.

[6] C. M. Orban and R. M. Teeling-Smith. 2020. Computational Thinking in Introductory Physics. *The Physics Teacher* 58, 4 (04 2020), 247–251. https://doi.org/10.1119/1.5145470 arXiv:https://pubs.aip.org/aapt/pte/article-pdf/58/4/247/9871960/247_1_online.pdf

[7] Emmanuel Schanzer. 2015. *Algebraic Functions, Computer Programming, and the Challenge of Transfer.* Ph. D. Dissertation. Harvard University.

[8] Osvaldo Soto. 2023. Discrete Math Project Collaborative. https://discretemathproject.net/.

[9] NGSS Lead States. 2013. Next Generation Science Standards: By states, for states. *The National Academies Press* (2013).

[10] Stanford University. 2023. Data Science - youcubed. https://www.youcubed.org/resource/data-literacy/.

[11] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2016. Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology* 25, 1 (2016), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Understanding the Leadership Structure and Mentoring Model of an Extracurricular Robotics Team

Key Findings from a Case Study

Hitender K Oswal
University of Washington
hitender@uw.edu

Jonathan Wick
Western Washington University
wickj2@wwu.edu

Seth Tandon
University of Southern California
sethtand@usc.edu

Ashley Brewster
Science and Math Institute
2.brewster.a@gmail.com

Sushil K Oswal
University of Washington
oswal@u.washington.edu

ABSTRACT

This study examines the leadership structure of an extracurricular robotics team where students from four high schools manage the team's work. Using a qualitative survey tool, it explores the differing role of mentors in student-led teams and investigates the impact of a student mentor model on team success, challenges faced, and skills learned. Preliminary findings indicate that the mentors for this student-led team play a key role in decision making; however, they use a hands-off style to steer students and their leads in the right direction. The study provides insights into how students perceive the role of team mentors and the actual role played by these mentors. A distinguishing feature of this study is the participation of four student members of this team in the design and implementation of this study.

ACM Reference format:

Hitender K Oswal, Jonathan Wick, Seth Tandon, Ashley Brewster, and Sushil K Oswal. 2024. Understanding the Leadership Structure and Mentoring Model of an Extracurricular Robotics Team: Key Findings from a Case Study. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024).* ACM, New York, NY, USA. 2 pages. https://doi.org/10.1145/3626253.3635603

1 BACKGROUND AND RELATED WORK

For broadening the participation of underrepresented groups in computing, researchers have explored the role of extracurricular robotics competitions in women and minority student's career choices [1, 2]. However, few studies focus on the leadership structure of these groups [3, 4].

Whereas literature notes the role of peer-to-peer learning, detailed studies about the guidance approaches employed by mentors are scarce [5, 6]. To understand students' learning processes, this study examines the mentor roles and guidance approaches of an extracurricular robotics team (RT) in one U.S. school district where students from 9th to 12th grade play lead roles in carrying on the work of various sub-teams—build, programming, scouting, business, outreach, media, awards, safety, and general management. RT has a hierarchical Student Mentorship Model (SMM) to train students in all facets of the team: student management coordinates all aspects of the team and supports sub-teams, whereas student leads manage individual sub-teams and often teach members. Along with SMM, RT has institutionalized a more direct peer-to-peer learning structure called the "Jedi-Padawan" system where experienced members (Jedis) teach new members (Padawans) essential skills for functioning in RT while also passing on the group's collective knowledge to the next generation. Using an open-ended qualitative survey, this study explores two key research questions about the workings of the group: RQ1) In this student-led robotics team, how is the role of mentors different from those in the adult mentor-led teams? and RQ2) how does this team structure affect student learning, challenges faced, and skills acquired?

Publications on broadening the participation of underrepresented student groups—women, ethnic minorities, and students with disabilities—through robotics competitions have consistently grown in computing and allied fields [2, 7]. In a survey of high school students, [4] found that students who perceive adult support more positively also rate their own leadership skills likewise. In two of the most relevant studies for this paper, researchers observed teams participating in FIRST Competitions and claimed that more mentor involvement meant students experienced an apprenticeship-based team structure while a hands-off approach led to students taking more initiatives [5, 6]. This study extends the existing work by analyzing RT and how its members view mentorship, student leadership, and learning.

2 METHODS

All researchers collaboratively formulated the two research questions for this study, developed survey questions, and coded the responses. The student researchers promoted survey

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635603

participation at team meetings and on the team's communication channel.

Out of the 51 registered members of RT, 32 regularly participate and 24 completed the survey. The 4 student researchers did not complete the survey to avoid conflict of interest. All data collection was anonymous. Survey responses were first inductively coded using Corbin and Strauss's Open Coding technique followed by a phase of Axial Coding [8]. These codes were reviewed in light of the research questions and survey responses to arrive at the key themes. While the results from this case study are not generalizable due to its small sample size, they provide a close-up view of RT's leadership structure from the students' perspective. Aggregated results of similar studies in the future could offer a broader picture about the leadership structure and mentoring roles of extracurricular robotics teams.

3 RESULTS

Preliminary findings indicate that the adult mentors—teachers functioning outside the school structure, industry professionals, and team alumni—in this student-led team play a major role in macro-level decision making while leaving the day-to-day leadership and management in student hands. Additionally, they use a hands-off, behind the scenes approach to steer students in the right direction, answer their questions, and fill the knowledge gaps in this otherwise autonomous-looking group [5]. The results also reveal that most student discourse focuses on peer-to-peer learning and technical work—robot design, manufacturing, and programming—with little conversation about the social aims of the group about broadening participation in computing.

When asked about their thoughts on the mentorship system, 66.7% of students preferred student-led learning with another 20% preferring a hybrid system. Only one response indicated that they would prefer a mentor learning system. Conversely, when asked how they felt about the current mentor involvement 45% of students were satisfied with the current system while 29% desired more mentor involvement.

Students attributed their learning in RT to several processes. While most students agreed that peer-to-peer, hands-on, and independent learning play a role, some attributed their learning to project-based learning.

When asked about their opinions on SMM in different contexts, there were a variety of responses. Many students believed that SMM improved communication skills and task management; was more enjoyable and taught more applicable skills; and 70% indicated that SMM was more motivating than mentor-lead models. Conversely, some responses indicated that SMM was no different or had negative effects on productivity. Some noted that the organization and planning was worse. In addition, there were comments about increased distractions and chaos within RT.

4 CONCLUSION

This study contributes to the body of knowledge on leadership models in extracurricular afterschool robotics teams, particularly on how student leads and adult mentors perceive and enact their mentoring roles. These initial findings reveal three major work characteristics in the case team: a hands-off role played by adult mentors (RQ1), a hierarchical leadership structure in spite of agency for students and their leads, and a strong tradition of peer-to-peer learning through the institutionalized Padawan and Jedi structure that students associate with agentic capability (RQ2). These findings about mentoring and leadership structure provide significant insights into the collaborative aspects of this team's functioning that give high school students spaces outside the classroom to develop their own mechanisms for collaboration, learning, and leadership while shedding light on how these mechanisms influence student interactions with one another and their adult mentors, develop a sense of agency among themselves, and offer explanation why afterschool robotics teams result in greater STEM participation downstream.

ACKNOWLEDGMENTS

This study went through the IRB review process at the University of Washington. All students participating in this study completed a permission form. In the case of a minor, the form was completed by a parent or legal guardian.

REFERENCES

[1] Stephanie Ludi. 2012. Educational robotics and broadening participation in STEM for underrepresented student groups. In Bradley S. Barker, Gwen Nugent, Neal Grandgenett, and Viacheslav Adamchuk (Eds.), *Robots in K-12 Education: A New Technology for Learning*. IGI Global, Hershey, PA.

[2] Tatjana Meschede, Zora Haque, Marji E. Warfield, Alan Melchior, Cathy Burack, and Matthew Hoover. 2022. Transforming STEM outcomes: Results from a seven-year follow-up study of an after-school robotics program's impacts on freshman students. *School Science and Mathematics* 122, 7, 343-357. https://doi.org/10.1111/ssm.12552

[3] Kathleen Morgan, Bradley Barker, Gwen Nugent, and Neal Grandgenett. 2019. Educational Robotics as a Tool for Youth Leadership Development and STEM Engagement. In Alpaslan Sahin and Margaret J. Mohr-Schroeder (Eds.), *STEM Education 2.0: Myths and Truths - What Has K-12 STEM Education Research Taught Us?* Brill, Boston, MA, 248-275.

[4] Donna Hancock, Patricia H. Dyk, and Kenneth Jones. 2012. Adolescent Involvement in Extracurricular Activities: Influences on Leadership Skills. *Journal of Leadership Education* 11, 1, 84-101.

[5] Nathan R. Dolenc, Claire E. Mitchell, and Robert H. Tai. 2016. Hands off: Mentoring a student-led robotics team. *International Journal of Science Education, Part B* 6, 2, 188-212.

[6] Nathan R. Dolenc, Robert H. Tai, and Douglas Williams. 2020. Excessive Mentoring? An Apprenticeship Model on a Robotics Team. *Journal of Research in STEM Education 6*, 2, 91–114. https://doi.org/10.51355/jstem.2020.81

[7] Michelle Craig and Diane Horton. 2009. Gr8 designs for Gr8 girls: a middle-school program and its evaluation. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE '09).* Association for Computing Machinery, New York, NY, USA, 221–225. https://doi.org/10.1145/1508865.1508949

[8] Juliet Corbin and Anselm Strauss. 2014. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (4th. ed.). Sage Publishing, Thousand Oaks, CA.

Linguistic Pedagogical Approaches to Transfer in Computer Science

Rosalind Owen
rowen@wested.org
WestEd
San Francisco, CA, USA

ABSTRACT

More students are encountering computer science at multiple grade levels and so are learning more than one programming language. There is an ever-growing body of research describing how students transfer knowledge from one language to another. Current research shows that transfer helps students learn a second programming language in the interim and improves attitudes and retention of students in computer science [2, 12]. While novice programmers generally struggle with the same concepts [1, 12], the exact difficulties and benefits of the transition to a second programming language differ depending on the programming languages the student is learning. In order to best serve students of different backgrounds and maintain their interest in the subject, the details of transfer for different programming language combinations must be understood. This poster surveys and analyzes the current research on transfer and provides a summary of the variety of challenges and advantages students face in learning a second programming language. Additionally, interdisciplinary pedagogical approaches are discussed, integrating perspectives from applied linguistics as novel solutions to the specific issues faced in programming language transfer.

ACM Reference Format:
Rosalind Owen. 2024. Linguistic Pedagogical Approaches to Transfer in Computer Science. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635566

1 BACKGROUND AND OVERVIEW

As more students take computer science courses at different points in their education, they are likely to encounter more than one programming language. Recent research has investigated analogical transfer, or how students use shared relationships to transfer knowledge from one context to another, in this case, from one programming language to another [9]. When students make connections between surface similarities that reflect real similarities, this is called positive transfer. Negative transfer occurs when these surface similarities lead to incorrect assumptions. In order to better support students in their transition from one language to another, researchers are describing how students manage a new programming language based on their previous programming language, along with various interventions meant to support positive transfer. The specific challenges students face while learning a new programming language vary depending on their prior experience [7], as do their strengths when transferring knowledge from their first programming language [11, 12].

Applied linguistics as a field has much research on the subject of transfer, specifically from one natural language to another. Both computer science education and natural language learning share the issue of how the syntax and semantics of one language influence the acquisition and use of another language. As is the case with programming languages, natural languages often have semantic distinctions or syntactic structures not present in another language. Researchers in applied linguistics have developed pedagogical solutions to help students overcome obstacles in language acquisition and leverage previous knowledge to understand the intricacies of each language's syntax and semantics. Interventions for avoiding negative semantic transfer have been studied at the vocabulary [8] and sentence [4] levels, which can be applied to learning keywords and syntax in computer science classrooms. Additionally, contextualizing languages and raising awareness of their characteristics can promote critical thinking about how to use each language and how their distinctions may be useful [5]. This approach can be applied to programming languages and be used to empower students in their own learning.

This poster reviews studies of transfer in computer science education and presents frequently encountered issues. Approaches from natural language learning are presented as novel solutions to common problems with second programming language learning in the computer science classroom.

2 TRANSFER BY PROGRAMMING LANGUAGE

Each programming language pairing is compared in a table that summarizes the instances of positive and negative transfer. The peculiarities of each pairing are explored, as well as generalizations based on shared characteristics between programming languages. An example follows below.

Students with experience in Scratch who are learning Python struggle with understanding which characteristics of loops apply to their prior language and which apply to the new language [10]. These students use conditional statements with the opposite condition than needed for while loops, an instance of negative transfer from Scratch repeat loops. Additionally, they sometimes handle the end of a while loop with an if statement to break the loop, rather than relying on the conditional statement, which is likely a negative transfer from Scratch forever loops.

Students also struggle to grasp concepts related to Python-specific traits [10]. There is often confusion about type matching, with students using operators on variables of different types, and type

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635566

casting, with students recasting data types with still the wrong type. Considering Scratch is a weakly typed language and Python is strongly typed, students may need more information to grasp this new concept. Students also did not understand how to initialize an empty list, instead initializing a list with one item in it and then later clearing it. This misconception likely stems from the fact that lists are not explicitly initialized in Scratch.

Transitions from Alice, Snap!, and Scratch to Java, as well as Python to Java, are also included in the poster.

3 APPROACHES FROM NATURAL LANGUAGE LEARNING

This section reviews pedagogical approaches to semantic transfer in natural languages, including perspectives from translanguaging [5] and cognitive linguistics [4]. The main contribution of this poster is the application of these approaches to the transfer issues observed in different programming language pairings and how these approaches can help computer science instructors.

Pedagogical translanguaging suggests that "intentional instructional strategies that integrate two or more languages" should be used in a multilingual classroom [3]. Pedagogical translanguaging can be incorporated into the classroom by raising metalinguistic awareness, or awareness of a language system (e.g., syntactic structures or sound patterns) [5]. In natural languages, transfer is often seen at the morphological level. Negative transfer occurs in the form of "false friends," or words that mean different things but look similar. For instance, students learning Spanish used the Spanish "carpeta" (English: "folder") as if it meant "carpet" in English (Spanish: "alfombra") [6]. However, students with more metalinguistic awareness were able to use their knowledge of the morphology of Spanish to make complex changes to the suffixes and roots of known words to guess new Spanish words in their writing (e.g., English "stressed" to Spanish "estresado") [6].

In the context of a computer science classroom, students' prior knowledge of any programming language would be bolstered by instruction that raises metalinguistic awareness. Their prior coding experience would be an asset that they could combine with metalinguistic awareness to approach novel coding situations. For example, students learning Python with experience in Scratch often use an if-statement to break from a conditional loop [10]. This confusion may be akin to the issue of "false friends." Students see syntax in Python related to an infinite loop and connect it with a forever loop in Scratch, not fully aware of the semantic importance of the conditional statement. With pedagogical translanguaging, the distinctions between finite and infinite loops, as well as conditional and unconditional loops, would be explained, so that students have the vocabulary with which to think about different kinds of loops. With more explicit conceptual knowledge about the forms that loops can take, students could approach new keywords and syntax in a more informed and analytical manner.

In the poster, two more methods, transcodification and input manipulation, are applied to other examples of transfer in computer science. In transcodification activities, students are shown equivalent constructions in two languages, and their literal and figurative meanings are discussed to explore subtle semantic differences [4]. For instance, students could compare pairs of equivalent loops in two programming languages, using flow-charts or diagrams as references to discuss semantic differences. Input manipulation also addresses confusion around the semantics of a particular structure, giving students positive evidence in context for the correct interpretation [4]. While a new control structure is being introduced, students could be shown code with key words highlighted, i.e. syntax highlighting, to illustrate the crucial differences between the languages and provide positive examples of the correct, contextualized use of the structure. Example applications of both of these methods are included in the poster.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2201209. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Amjad Altadmri and Neil C.C. Brown. 2015. 37 million compilations. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (2015). https://doi.org/10.1145/2676723.2677258

[2] Michal Armoni, Orni Meerbaum-Salant, and Mordechai Ben-Ari. 2015. From scratch to "real" programming. *ACM Transactions on Computing Education* 14, 4 (2015), 1–15. https://doi.org/10.1145/2677087

[3] Jasone Cenoz and Durk Gorter. 2020. Teaching English through pedagogical translanguaging. *World Englishes* 39, 2 (2020), 300–311.

[4] Paolo Della Putta. 2016. Do we also need to unlearn constructions? the case of constructional negative transfer from Spanish to Italian and its pedagogical implications. *Applied Construction Grammar* (2016), 237–268. https://doi.org/10.1515/9783110458268-010

[5] Carles Fuster. 2022. Lexical transfer as a resource in pedagogical translanguaging. *International Journal of Multilingualism* (2022), 1–21. https://doi.org/10.1080/14790718.2022.2048836

[6] Carles Fuster. 2022. *Lexical transfer in pedagogical translanguaging: Exploring intentionality in multilingual learners of Spanish.* Ph. D. Dissertation. Department of Education, Stockholm University.

[7] Ryan Garlick and Ebru Celikel Cankaya. 2010. Using Alice in CS1. *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (2010). https://doi.org/10.1145/1822090.1822138

[8] Nan Jiang. 2004. Semantic Transfer and its implications for vocabulary teaching in a second language. *The Modern Language Journal* 88, 3 (2004), 416–432. https://doi.org/10.1111/j.0026-7902.2004.00238.x

[9] Yvonne Kao, Bryan Matlen, and David Weintrop. 2022. From one language to the next: Applications of analogical transfer for programming education. *ACM Transactions on Computing Education* 22, 4 (2022), 1–21. https://doi.org/10.1145/3487051

[10] Majeed Kazemitabaar, Viktar Chyhir, David Weintrop, and Tovi Grossman. 2023. Scaffolding Progress: How Structured Editors Shape Novice Errors When Transitioning from Blocks to Text. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 556–562. https://doi.org/10.1145/3545945.3569723

[11] David Weintrop, Connor Bain, and Uri Wilensky. 2017. Blocking Progress? Transitioning from Block-based to Text-based Programming. In *Proceedings of the American Educational Research Association*. 1–8.

[12] David Weintrop and Uri Wilensky. 2019. Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers amp;amp; Education* 142 (2019), 103646. https://doi.org/10.1016/j.compedu.2019.103646

Almost Correct Solvers for Combinational Logic Assignments

Sunjae Park
parks6@wit.edu
Wentworth Institute of Technology
Boston, USA

ABSTRACT

AlmostCorrect is an educational tool for working with combinational logic. Combinational logic can have several different, but correct solutions, and many more solutions that are almost, but not quite, correct. However, existing solutions typically generate one correct solution. AlmostCorrect instead generates all possible correct solutions and all solutions that have minor mistakes, such as missing a simplification step or reading the table incorrectly. These "almost" solutions can be used to either provide distractors for multiple choice questions or as a rubric for assigning partial credit.

CCS CONCEPTS

• **Hardware → Combinational synthesis**; • **Social and professional topics → Student assessment**.

KEYWORDS

Autograding;Digital Logic

ACM Reference Format:
Sunjae Park. 2024. Almost Correct Solvers for Combinational Logic Assignments. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635539

1 INTRODUCTION

As part of the System Fundamentals knowledge area, the CS 2013 Task Force recommends [8] that all computer science students be able to describe the basic building blocks of computers and be able to design simple logic circuits [10, 12]. However, one-to-one translation results in an inefficient circuit. So a crucial step is to *minimize* the logic expression before laying out the circuit.

Unfortunately, grading assignments that assess logic expression minimization can be arduous. Due to the nature of these expressions, there can be many correct answers. There are many symbols involved which require careful reading. To make matters worse, the number of answers that are correct, but not *fully* minimal are even more numerous.

AlmostCorrect is a tool that helps instructors with these assessments. It does so in two ways: 1) given a truth table and a student answer, it can automatically determine of the provided student answer is the correct minimal expression or not, and 2) given a truth table, it can generate distractors for multiple choice questions. Using this, an instructor can grade more accurately and save time.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635539

2 BACKGROUND ON LOGIC EXPRESSION MINIMIZATION

When building logic circuits, students are given either a description of the logic expression, from which they need to first generate a truth table, or given the truth table directly. A truth table is a table that lists all possible input combinations of 0s and 1s, and the corresponding output value.

Once a truth table has been determined, the truth table can be converted using either laws of Boolean algebra, or by using a Karnaugh map (aka K-map). Unfortunately, there can be multiple solutions. One way multiple solutions can show up is because logical AND and logical OR operations are commutative. This means that the following expressions are identical:

- $\overline{A}B + BC$
- $CB + \overline{A}B$
- $BC + B\overline{A}$

...and so on. As can be seen above, due commutativity, there is an exponential explosion of correct solutions. This makes grading time consuming, and increases the possibility of mistakes. Even if variables need to be written in alphabetical order, due to NOT operators this will still lead to several different solutions.

In addition, sometimes the output can be recorded as a "don't care", commonly recorded using the letter "X". Some input combinations are supposed to be impossible, maybe because it's part of a finite state machine and the state transition logic precludes it from happening. In this case, the output for the said input can be *either* 0 or 1. This can also lead to multiple correct solutions. For example, let's say we have the following truth table:

A	0	0	1	1
B	0	1	0	1
Q	X	1	0	X

There are two correct solutions to the above truth table; $\overline{A}$ or B.

3 DESCRIPTION OF ALMOSTCORRECT

AlmostCorrect is a tool that helps instructors with grading logic expression minimization problems. It does so by 1) automatically grade them and provide explanations as to why (if so) the solution was not fully correct, and 2) generate partial solutions for use in multi-choice questions as a distractor.

In the first mode, AlmostCorrect's autograder expects two inputs: the truth table and the proposed student solution. The first step it takes is to analyze the truth table and generate the correct solution(s). The truth table contains for each input combination the corresponding output, which can be either 0, 1, or an X (for *don't*

cares). It then uses the Quine-McCluskey Method [9] to minimize the solution(s).

Once a set of correct, minimal solutions are generated, AlmostCorrect then investigates if the proposed solution P can be found in this set. It first sorts the terms of P (so that duplicates can be matched), and searches the set. If a sorted version of P, which we call T is found, AlmostCorrect's job is done, and P is a fully *correct and minimal* solution.

However, if the proposed solution P does not exist, AlmostCorrect then checks if it is at least functionally equivalent to the truth table. Functionally equivalent in this case means all 0s and 1s of T result in 0s or 1s respectively in P. Any don't cares in T do not matter. If so, P is a *correct but not minimal* solution.

Next, we check for incorrect handling of don't cares. A different logic expression can result depending on how the don't care values are assigned (to either 0 or 1). Although some of these expressions can be minimal, others are not. This step generates all possible combinations of truth tables with don't care terms set to either 0 or 1. For each concrete version of T, the autograder checks if the answer matches. If so, the answer is *Incorrect don't-cares*.

If all of the above attempts fail, the autograder marks the solution as *incorrect*.

In the second mode, AlmostCorrect can choose one category of answers, and randomly choose a solution from the said category. For example, it can be directed to generate a *correct but not minimal* solution.

4 RESULTS

AlmostCorrect has been used by the instructor for grading and for generating distractor questions for a multiple choice question. With traditional logic expression solvers, only one answer from the *fully correct and minimal* answer was generated. Without manual intervention, the instructor had to mark all other answers as incorrect, although the students only had minor mistakes. By using the tool, the instructor was able to provide partial credit in the final exam, which addresses one student concern with automated grading [5, 7, 14].

5 RELATED WORK

The topic of minimizing logic functions is a well studied area. Many textbooks introduce the Karnaugh map (aka K-map), a graphical method where students identify rectangles of 1s. These rectangles can be used to create minimal Boolean terms (aka "minterms") that make up the minimal logic expression. This method works well for logic expressions with up to 4 variables [9, 12].

To generate minimal solutions automatically, the Quine-McCluskey method, or tabulation method [2, 11] can be used. Unlike the previous method, this method uses a table to minimize the expression, and does not have a limit to the number of variables. This method is mainly used to programatically minimize logic expressions [1, 4]. However, these tools only generate a *single fully minimal* answer.

JLS [13] and Logisim [3, 6] are digital circuit simulation and design tools. They allow students to visually lay out digital logic components and connect them in a simulated environment. Part of its functionality is automatic generation of truth tables, Karnaugh maps, and the corresponding logic expression from a given logic circuit. However, these tools are mainly concerned with correctness, but not with minimization.

As a core learning outcome, students should be able to do some logic circuit design and minimize them. However, the nature of logic circuit design results in multiple solutions. AlmostCorrect is a tool that targets this learning objective, by assisting in grading and providing several possible solutions for logic expression minimization questions. By providing all possible solutions, and answers that are correct but not fully minimal, AlmostCorrect reduces instructor burden in grading, and allows for more flexibility.

6 CONCLUSION

This poster describes AlmostCorrect, a command-line tool that helps instructors with work. Given a truth table and a potential solution, it generates a minimal solution and checks if the proposed solution matches. If not, it then tries a variety of transforms to check if the proposed solution lands in other answer categories. By trying these additional solutions, it is able to reduce time consumed in grading logic expression minimization problems, and help the instructor provide additional feedback to the students.

REFERENCES

[1] 32x8. [n. d.]. *Online Karnaugh map solver with circuit for up to 6 variables.* Retrieved Aug 7, 2023 from http://www.32x8.com/index.html

[2] N.N. Biswas. 1971. Minimization of Boolean Functions. *IEEE Trans. Comput.* C-20, 8 (1971), 925–929. https://doi.org/10.1109/T-C.1971.223373

[3] Carl Burch. 2002. Logisim: A Graphical System for Logic Circuit Design and Simulation. *J. Educ. Resour. Comput.* 2, 1 (mar 2002), 5–16. https://doi.org/10.1145/545197.545199

[4] Charlie Coleman. [n. d.]. *Karnaugh Map Solver.* Retrieved Aug 7, 2023 from https://www.charlie-coleman.com/experiments/kmap/

[5] Marjorie Darrah, Edgar Fuller, and David Miller. 2010. A Comparative Study of Partial Credit Assessment and Computer-Based Testing. *Journal of Computers in Mathematics and Science Teaching* (01 2010).

[6] Logisim evolution developers. [n. d.]. *Logisim Evolution.* Retrieved Aug 7, 2023 from https://github.com/logisim-evolution/logisim-evolution

[7] Georgiana Haldeman, Andrew Tjang, Monica Babeş-Vroman, Stephen Bartos, Jay Shah, Danielle Yucht, and Thu D. Nguyen. 2018. Providing Meaningful Feedback for Autograding of Programming Assignments. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 278–283. https://doi.org/10.1145/3159450.3159502

[8] Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* Association for Computing Machinery, New York, NY, USA.

[9] M. Morris Mano. 1991. *Digital Logic and Computer Design* (3rd. ed.). PrenticeHall.

[10] Suzanne J. Matthews, Tia Newhall, and Kevin C. Webb. 2021. Dive into Systems: A Free, Online Textbook for Introducing Computer Systems. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 1110–1116. https://doi.org/10.1145/3408877.3432514

[11] E. J. McCluskey. 1956. Minimization of Boolean functions. *The Bell System Technical Journal* 35, 6 (1956), 1417–1444. https://doi.org/10.1002/j.1538-7305.1956.tb03835.x

[12] Yale Patt and Sanjay Patel. 2020. *Introduction to Computing Systems* (3rd. ed.). McGrawHill.

[13] David A. Poplawski. 2007. A Pedagogically Targeted Logic Design and Simulation Tool. In *Proceedings of the 2007 Workshop on Computer Architecture Education* (San Diego, California) *(WCAE '07)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/1275633.1275635

[14] Seth Poulsen, Shubhang Kulkarni, Geoffrey Herman, and Matthew West. 2023. Efficient Feedback And Partial Credit Grading For Proof Blocks Problems. In *Artificial Intelligence in Education: 24th International Conference, AIED 2023, Tokyo, Japan, July 3–7, 2023, Proceedings* (Tokyo, Japan). Springer-Verlag, Berlin, Heidelberg, 502–514. https://doi.org/10.1007/978-3-031-36272-9_41

The Potential of Humanitarian Applications to Increase Interest and Motivation of Underrepresented Student Groups

Lori Postner
Nassau Community College
Garden City, NY, USA
lori.postner@ncc.edu

Heidi J. C. Ellis
Western New England University
Springfield, MA, USA
ellis@wne.edu

Gregory W. Hislop
Drexel University
Philadelphia, PA, USA
hislop@drexel.edu

Wesley Shumar
Drexel University
Philadelphia, PA, USA
shumarw@drexel.edu

ABSTRACT

This study investigated whether humanitarian applications of computing are more interesting and motivating to students than non-humanitarian options and whether there are differences across race/ethnic groups. A survey was developed to measure students' interest by asking them to choose between two applications, one humanitarian and one non-humanitarian. Students were also asked to rate how motivating they would find working on a collection of humanitarian and non-humanitarian applications.

376 introductory computing students at four institutions completed the survey and 340 student responses were analyzed based upon their self-reported races/ethnicities. Students in all race/ethnic groups were more interested in and motivated by humanitarian applications over non-humanitarian options. Black, Latinx and White students chose humanitarian applications more often than Asian students, indicating a higher level of interest in computing for social good. All four groups rated humanitarian applications statistically higher than non-humanitarian at the p<.001 level.

CCS CONCEPTS

• **Social and professional topics → Race and ethnicity**.

KEYWORDS

computing for social good, diversity in computing, HFOSS

ACM Reference Format:
Lori Postner, Heidi J. C. Ellis, Gregory W. Hislop, and Wesley Shumar. 2024. The Potential of Humanitarian Applications to Increase Interest and Motivation of Underrepresented Student Groups. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635513

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635513

1 INTRODUCTION

Prior work has found evidence that humanitarian applications of computing are particularly motivating to women and other traditionally underrepresented groups of students [1]. The authors' prior work looked at students taking upper-level computing courses. Unfortunately, the impact on underrepresented groups is difficult to measure statistically as these courses had few students from these groups. This study looks specifically at interest in and motivation of humanitarian aspects of computing across gender and race/ethnic groups in introductory classes which have a greater enrollment and diversity. This poster addresses the research question:

RQ: Do students across race/ethnic groups prefer humanitarian applications over non-humanitarian?

Interest has long been known as a motivational variable and is defined as "the state of engaging or the predisposition to reengage with particular classes of objects, events, or ideas over time" [5] p. 112. Interest is critical to academic success as it stimulates students to engage in thinking about a topic, to understand and organize the problems related to the topic and to persist when solving these problems. This persistence eases the feeling of effort required to solve problems and increases the possibility of successful problem solution [7]. Stimulating interest holds the potential to increase persistence of underrepresented students in computing.

2 METHODS

A review of the survey used in prior work [2] concluded that the technologies implied by the survey choices might be a confounding factor. As a result, the survey was revised to make the technologies used consistent across applications. The instrument used in this study included questions from CRA's Data Buddies Survey [3] and demographic information and details are provided in [6]. Part 1 of the survey focused on student interest and asked **Would You Rather** questions. Students were provided with six applications applied to two different domains and were asked to select between the domain with a humanitarian focus and the domain with the non-humanitarian focus.

Part 2 of the survey focused on student motivation and questions are referred to as **Motivation to Learn Programming** questions. This section contained eight different applications and students were asked to rate their motivation to work on these applications based on a 5-point Likert scale.

Table 1: Survey Participants by Gender and Race/Ethnicity W (Woman), M (Man), NB (Non-binary), ND (Did not disclose/Self-described) and Tot (Totals)

Race/Ethnicity	W	M	NB	ND	Tot
Asian	**16**	**48**	0	0	46
Black/African American	**10**	**34**	2	0	46
Latinx/Hispanic	**15**	**30**	1	1	47
White	**56**	**121**	4	2	183
Totals	**103**	**254**	**7**	**5**	**369**

Table 2: One-tailed, two-sample t-tests and Cohen's d scores of Asian student interest in humanitarian applications vs other race/ethnic groups.

Race/ Ethnicity	T-test	Cohen's d	Effect Size
Black	t(111)=2.10, p<.05	0.3366	SM
Latinx	t(110)=1.72, p<.05	0.4065	SM
White	t(247)=2.59, p<.05	0.3806	SM

3 DATA ANALYSIS

The study involved four academic institutions: a community college, a small liberal arts college, a private university, and a public state university between fall 2020 and spring 2021. Participants were elicited from students in undergraduate introductory computing courses and were typically in their first or second year of study. 376 students participated in the survey. Of these, only four groups were of sufficient size to support further analysis: Asian, Black or African American, Latinx or Hispanic, and White. Table 1 shows the breakdown of gender and race/ethnicity of the survey participants.

Across all respondents, humanitarian options were chosen more often and rated higher overall than non-humanitarian options. The analysis of the **Would You Rather** questions showed students from all groups chose the humanitarian option more than the non-humanitarian option, indicating their interest in the social good aspect of computing.

Asian students chose the humanitarian option 61.6% of the time whereas the other groups chose the humanitarian option more than 70% of the time (Black-70.6%, Latinx-72.3%, White-71.3%). The differences between Asian students' interest and Black, Latinx and White students' interest was statistically significantly at the p<.05 level based upon one-tailed, two-sample t-tests (Table 2) demonstrating that Asian students were less interested in the humanitarian options than the other race/ethnic groups and the Cohen's d shows a small to medium effect size.

All race/ethnic groups rated the humanitarian options significantly higher than the non-humanitarian options for the **Motivation to Learn Programming** questions, at the p<.001 level based upon one-tailed, paired t-tests (Table 3). The largest differentials between the ratings were in the Latinx (1.08) and White (0.95) groups whereas the Asian (0.73) and Black (0.69) ratings were closer. The Cohen's d indicated a large effect size for the Asian, Latinx and White groups and a medium to large effect size for the Black group.

Table 3: One-tailed, paired t-tests and Cohen's d scores demonstrating humanitarian options were rated higher than the non-humanitarian options for the motivation questions.

Race/ Ethnicity	T-test	Cohen's d	Effect Size
Asian	t(64)=7.13, p<.001	0.8653	L
Black	t(44)=4.24, p<.001	0.6972	ML
Latinx	t(45)=7.76, p<.001	1.1806	L
White	t(176)=11.81, p<.001	1.0536	L

4 DISCUSSION

The results above demonstrate that students in all race/ethnic groups prefer humanitarian computing examples in their courses, supporting **RQ:** Do students across race/ethnic groups prefer humanitarian applications over non-humanitarian? Such preferences are an indicator choosing application areas that demonstrate the ability of computing to help others may be useful in engaging students from all race/ethnic groups regardless of gender. Making the social impacts of computing visible can be as simple as re-framing existing assignments with a humanitarian focus [4] or as comprehensive as having students contribute to a humanitarian open source project [1, 2].

ACKNOWLEDGMENTS

Some survey questions were obtained from the Computing Research Association's (CRA) Data Buddies Survey which was made possible by National Science Foundation (NSF) grants CNS-1246649, DUE-1431112, and/or DUE 1821136 awarded to CRA. This study was supported under Grants DUE 1225738, 1225688, 1225708, 2012966, 2013069, 2012979, 2012999, and 2012990. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of NSF or CRA.

REFERENCES

[1] Heidi J. C. Ellis, Gregory W. Hislop, Stoney Jackson, and Lori Postner. 2015. Team Project Experiences in Humanitarian Free and Open Source Software (HFOSS). *ACM Trans. Comput. Educ.* 15, 4, Article 18 (dec 2015), 23 pages. https://doi.org/10.1145/2684812

[2] Heidi J. C. Ellis, Gregory W. Hislop, and Lori Postner. 2021. The Power of Open Source for Social Good to Increase Diversity in Computing. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 1305. https://doi.org/10.1145/3408877.3439628

[3] Center for Evaluating the Research Pipeline (2021). 2021. *Data Buddies Survey 2019 Annual Report.* Technical Report. Computing Research Association, Washington, DC.

[4] Michael Goldweber. 2018. Strategies for Adopting CSG-Ed In CS 1. In *2018 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. 1–2. https://doi.org/10.1109/RESPECT.2018.8491703

[5] Suzanne Hidi and K. Ann Renninger. 2006. The Four-Phase Model of Interest Development. In *Educational Psychologist*, Vol. 41. 111–127. https://doi.org/10.1207/s15326985ep4102_4

[6] Lori Postner, Gregory W Hislop, and Heidi J.C. Ellis. 2023. Humanitarian Applications Increase Interest and Motivation of Women in Computing. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 416–422. https://doi.org/10.1145/3545945.3569832

[7] K. Ann Renninger and Suzanne Hidi. 2017. *The Power of Interest for Motivation and Engagement.* Routledge.

Immersing Undergraduates in Interdisciplinary Course Collaborations

S. Monisha Pulimood
Computer Science
The College of New Jersey
Ewing, NJ, USA
pulimood@tcnj.edu

Diane C. Bates
Sociology
The College of New Jersey
Ewing, NJ, USA
bates@tcnj.edu

Kim Pearson
Journalism & Prof. Writing
The College of New Jersey
Ewing, NJ, USA
kpearson@tcnj.edu

ABSTRACT

To address today's complex problems, professionals must work with people with domain expertise in a variety of disciplines and from different backgrounds. Undergraduates in computing programs have opportunities to work on team projects in courses, where typically, all students have similar expertise in the course content. However, they do not gain much experience in working on projects where team members are required to apply domain knowledge from different disciplinary backgrounds to effectively address complex societal problems. In the past four years, 176 students in a software engineering course at a public, primarily undergraduate institution, have collaborated on teams with students enrolled in business, statistics, and criminology courses to address issues identified by community partners. In evaluating their experience, students appreciated the interdisciplinary collaborative opportunities, despite the challenges and frustrations that come from perceptions of inequality in workload, and inconsistencies in team dynamics. Additionally, they consistently rated their own learning of course content higher at the end of the semester on multiple quantitative measures; this is corroborated by the students' grades on course assignments. The authors conclude that mentored and supportive environments, where professors offer explicit guidance on collaborating with people from diverse backgrounds and disciplines, and opportunities to practice, and sometimes fail, help prepare students for the problem-solving challenges they will encounter in the workplace. This poster reflects on the coursework, grading practices, students' learning outcomes, pedagogy, and guidance on developing interdisciplinary team skills.

ACM Reference format:
S. Monisha Pulimood, Diane C. Bates and Kim Pearson. 2024. Immersing Undergraduates in Interdisciplinary Course Collaborations. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635598

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635598

1 INTRODUCTION

Undergraduate curricula typically include courses that deepen disciplinary knowledge, as well as courses outside the majors that broaden students' exposure to other domains of inquiry. Some of these courses, particularly at the upper-level, may require students to work on team projects; these are usually focused on applying course concepts, and team members are assumed to have a common understanding of concepts. Non-major courses are often perceived by students to be irrelevant to their disciplinary areas, and students do not always comprehend how they can collaborate effectively with people from other disciplines to integrate and apply diverse knowledge areas and perspectives. Rapid advances in technology, such as development tools and infrastructure for hosting software applications, are enabling the development of software applications to address increasingly complex problems [3, 7, 10] whose solutions require integration of expertise from multiple disciplines. It is imperative that developers of these solutions gain experience in interdisciplinary collaboration. The disconnect, and lack of respectful and effective communication between experts from disparate disciplines frequently result in applications that are inadequate, do not meet users' requirements, or worse, lead to failures that have massive impact on lives and the economy. To better prepare students to face these challenges, computing programs must provide them with opportunities within the curriculum to learn how to collaborate with people from different backgrounds, and to integrate concepts from a variety of disciplines to design and implement software solutions. Students benefit from courses where they are immersed in interdisciplinary and collaborative experiences, with guidance from the instructors.

Much of the literature on course-based teamwork assumes that students work on discipline-specific projects. Interdisciplinary collaborative experiences discussed are generally in seminar courses or research-based projects. The authors have been experimenting with a pedagogical model that brings together faculty and students in two courses from different disciplines, and a community partner, to collaborate on a project that requires interdisciplinary knowledge to address a community-identified need [4, 11, 12]. This poster describes the pedagogy and guidance on developing interdisciplinary collaboration skills in Software Engineering, an upper-level required course, taught by Pulimood.

2 BACKGROUND

The Software Engineering (SE) course at The College of New Jersey (TCNJ) follows the ACM/IEEE [6] curriculum guidelines. Among other outcomes, SE aims to prepare students to design and implement a computing-based solution to meet requirements; function effectively on a team; and recognize professional responsibilities, and make informed judgments in computing practice based on ethical principles. These outcomes are aligned with ABET's criteria [1] for computing programs, the ACM Code of Ethics [2], and TCNJ's mission to develop critical thinkers and responsible citizens, who will contribute to, and enhance, their communities [14]. Collaborating courses and community partners varied. For example, in one semester, SE collaborated with Penology (offered by the Criminology department) and the Atlantic Center for Capital Representation [11].

3 IMMERSION IN INTERDISCIPLINARY COLLABORATION

In each semester, both courses in a collaboration were taught separately, in their own classrooms and time slots, enabling the instructors to cover disciplinary content. The instructors formed teams composed of six or seven students from both classes, intentionally balancing them by gender, class (e.g., junior or senior), and major, to promote equity and learning among students of different disciplines. Over the 15-week semester, the two classes and community partner came together on four to six occasions during class time for instructor-mediated teamwork, and teams met outside class time to complete projects.

The authors' experience is consistent with the findings of previous researchers about the importance of clearly defining projects, tasks, and roles, and ensuring project deliverables and milestones are realistic and achievable within allotted timeframes [8, 13]. The project goals drew from both courses equally, with emphasis on the interdisciplinary nature of the project. To alleviate perceptions of unfair distribution of work and responsibilities of each discipline, the project document specified the tasks for which team members from each class would take the lead. All team members were expected to understand the domain, and contribute to brainstorming ideas and designing modules. SE students were expected to focus on the technical (computational) aspects of design and development, while the students in the collaborating class were responsible for the domain content, and ensuring that the application met the needs of the community partner. Course-specific assignments provided scaffolding for students to become more proficient in course concepts, and to learn to communicate and collaborate with team members from the other class. For example, SE students delved deeply into the ACM Code of Ethics [2], considering principles like, "be fair and take action not to discriminate," in the context of products they will build, as well as in how they interact with their teammates. Each team developed a team code, inspired by the codes of conduct adopted by opensource communities [9] to promote respectful behavior and accountability towards each other. To emphasize collaboration over competition, and deep learning, creativity and risk-taking over following instructions [5], grades were not assigned to interim deliverables. Instead, teams reflected on the detailed feedback provided throughout the semester, and revised the project for the next iteration. The final grade for the team was based on criteria such as the value the project provided to the community partner, the collaboration within the team, and effort put into revising the project based on feedback.

4 FINDINGS AND ANALYSIS

Instructors' impressions and qualitative surveys of students at the end of each semester found that students appreciated these collaborative experiences despite some frustration resulting from perceptions of inequality in workload, and inconsistencies in team dynamics based on the personalities within the groups. Students felt that they were able to leverage their individual strengths in the creation of a meaningful solution. Responsibility and efficacy are necessary building blocks to address societal problems, such as systemic bias. While they encountered challenges, this experience reinforced to the faculty and their students, the value of interdisciplinary collaboration.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation (NSF), Award # 1914869. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF. Additional support was provided by TCNJ's Barbara Meyers Pelson Chair in Faculty-Student Engagement award (2018-2021).

REFERENCES

[1] ABET. 2023. Criteria for Accrediting Computing Programs. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2023-2024/. Retrieved 10/13/2023.
[2] ACM. ACM Code of Ethics and Professional Conduct. https://www.acm.org/code-of-ethics. Retrieved 10/13/2023.
[3] Andreessen, M., 2011. Why software is eating the world. Wall Street Journal, 20 (2011), p.C2.
[4] Bates, Diane C., Kim Pearson and S. Monisha Pulimood. 2022. Collaborating Across Boundaries: Innovative Pedagogy and the Difficulty of Quantitatively Assessing Change in Scientific Learning. 117th Annual Meeting of the American Sociological Association, Los Angeles, CA, August 5-9.
[5] Susan Blum and Alfie Kohn. 2020. Ungrading: Why Rating Students Undermines Learning (and What to Do Instead). (1st ed.), Morgantown: West Virginia University Press.
[6] CC2020 Task Force. Computing Curricula 2020: Paradigms for Global Computing Education. ACM, NY, NY, USA.
[7] Jacobson, I., Ng, P.W., McMahon, P.E. and Goedicke, M., 2019. The essentials of modern software engineering: free the practices from the method prisons! Morgan & Claypool.
[8] David A. Kolb. 2014. Experiential learning: Experience as the source of learning and development. Pearson Education. US.
[9] Organization for Ethical Source. Contributor Covenant. https://www.contributor-covenant.org/. Retrieved 10/13/2023
[10] Parker, K. (2020). Software still eating world. In Plant Engineering (Vol. 74, Issue 7, pp. 5–5). CFE Media LLC.
[11] Pulimood, S. M., Leigey, D. Bates, K. Pearson, and S. Mishra. 2022. A Project-Based Collaboration between Software Engineering and Criminology Students: Building Applications to Understand Racial Injustice in the Criminal Justice System. Proc. of the 44th Intl. Conf. on Software Engineering, Pittsburgh, PA: May 22-27.
[12] Pulimood, S.M., K. Pearson, K. and D.C. Bates. 2016. A Study on the Impact of Multidisciplinary Collaboration on Computational Thinking. In Proc. of the 47th ACM Tech. Symp. on CS Ed. (ACM SIGCSE), Memphis, TN, March 2-5.
[13] Dale H. Schunk. 2012. Learning Theories: An Educational Perspective, 6th Ed. Pearson. New York, NY.
[14] TCNJ. TCNJ's Mission. https://tcnj.edu/about/mission/. Retrieved 10/13/2023.

Student Perceptions of Authentic Learning to Learn White-box Testing

Akond Rahman
akond@auburn.edu
Department of Computer Science and Software Engineering, Auburn University
Auburn, Alabama, USA

Yue Zhang
yzz0229@auburn.edu
Department of Computer Science and Software Engineering, Auburn University
Auburn, Alabama, USA

Fan Wu
fwu@tuskegee.edu
Department of Computer Science, Tuskegee University
Tuskegee, Alabama, USA

Hossain Shahriar
hshahriar@uwf.edu
Center for Cybersecurity, University of West Florida
Pensacola, West Florida, USA

ABSTRACT

The pivotal role of white-box testing with respect to software quality assurance, necessitates dissemination of education materials related to white-box testing in the course curriculum. In this poster, we describe our experiences in conducting an authentic learning-based exercise related to white-box testing. From a conducted survey with 124 students, we observe the authentic learning-based exercise to be helpful for students to learn about white-box testing.

CCS CONCEPTS

• **Software and its engineering** → *Software verification and validation.*

KEYWORDS

authentic learning, white-box testing, student perception

ACM Reference Format:
Akond Rahman, Yue Zhang, Fan Wu, and Hossain Shahriar. 2024. Student Perceptions of Authentic Learning to Learn White-box Testing . In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635584

1 INTRODUCTION

Due to the importance of software testing with respect to software quality assurance [1], education-focused dissemination of testing techniques, such as white-box testing is crucial. White-box testing is a software testing technique that takes into account the internal mechanism of a software system [1]. Unlike black-box testing, in the case of white-box testing a software practitioner has visibility of the internal workings of a software system or a computer program [1]. Dissemination of education materials related to white-box testing can enable students to gather knowledge about white-box testing, and later apply the obtained knowledge in a professional setting.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635584

In this preliminary work, we describe students' perceptions about an exercise related to white-box testing. We develop the exercise using an authentic learning-based approach [4]. The exercise is deployed in a course taught at Auburn University (AU). Next, we use a survey to assess if students find the exercise helpful to learn about white-box testing.

2 RELATED WORK

Our paper is related with prior publications that have investigated instructional approaches to educate students on topics related to software quality assurance. Valle et al. [2] found game-based learning to be helpful for learning software testing. Rahman et al. [6] used authentic learning to educate students about secure development of Ansible manifests.

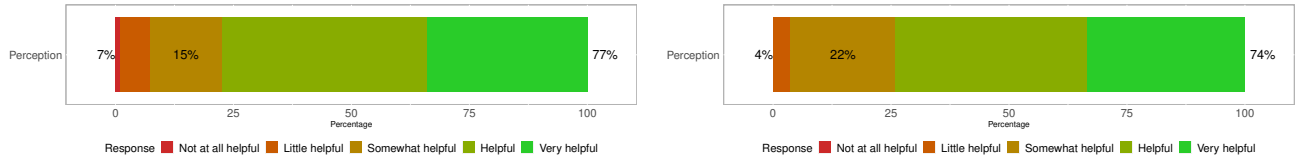
We observe existing research to study approaches to effectively disseminate topics related to software quality assurance in the course curriculum but a lack of research that focuses on instructional approaches to foster the learning of white-box testing. We address this research gap by investigating students' perceptions on authentic learning for learning about white-box testing.

3 METHODOLOGY

We take motivation from prior work [6] that used authentic learning for a topic related to infrastructure as code [5]. Our conjecture is that an authentic learning-based exercise can be helpful for students to learn about white box testing. We substantiate our conjecture by constructing an exercise using authentic learning, and then collecting student feedback for the conducted exercise. We *first* provide background on authentic learning. *Second*, we describe the conducted authentic learning-based exercise.

3.1 Authentic Learning

Authentic learning is an instructional approach that emphasizes on exposing students to real-world problem-based activities [3]. Curriculum modules developed using authentic learning include certain characteristics [4]: (i) focusing on hands-on exercises that have relevance to the real-world, (ii) allowing for students to have a diverse set of perspectives for the same exercise, and (iii) facilitating availability of resources to solve the exercises. An authentic learning-based exercise consists of three components: *first*, as part



a . Perceptions of undergraduate students

b . Perceptions of graduate students

Figure 1: Student perceptions of authentic learning to learn about white-box testing.

```
1 def divide(v1, v2):
2     temp =  v1 / v2
3     return temp
4
5 def fuzzValues():
6     '''
7     In-class Demonstration: expected software testing
8     '''
9     # checking if program works for 2 and 1
10    res = divide(2, 1)
11    '''
12    In-class Demonstration: unexpected software testing  to trigger
      ↪  a divison by zero exception
13    '''
14    # checking if program works for 2 and 0
15    res = divide(2, 0)
16    '''
17    Post-lab exercise: generate three more exceptions with the two
      ↪  inputs $v1$ and $v2$
18    '''
19    print(res)
20    print('='*100)
21
22 def simpleFuzzer():
23    fuzzValues()
```

Listing 1: Python program used to conduct the exercise.

of pre-lab content dissemination, necessary background is provided to students prior to conducting the exercise; *second*, as part of in-class experience, students are provided hands-on experiences within a classroom setting, so that students gain necessary hands-on experience by working on a problem that has real-world relevance; and *third*, as part of the post-lab activity, after the in-class experience is complete, students are assigned a post-lab activity, where the students solve a problem that is a variant of the in-class activity.

3.2 White-box Testing Exercise

The authentic learning-based exercise consists of three components: (i) pre-lab content dissemination, (ii) in-class demonstration, and (iii) post-lab exercise. *First*, as part of pre-lab content dissemination, the instructor provided necessary background on white-box testing. *Second*, as part of in-class demonstration, the instructor used a Python program that can perform division operations. The Python program is available in Listing 1. Availability of the program throughout the exercise enabled the instructor to demonstrate the internals of the program to the students. The instructor started with a simple implementation where `temp = v1 / v2` is used to perform division. Later, using the concept of white-box testing, the instructor demonstrated a 'Division By Zero' exception (line#11-#15). *Third*, as part of post-lab exercise the students are asked to generate three more exceptions with the two inputs $v1$ and $v2$ (line #16-#18). Once the exercise is completed, we ask the students to complete a survey, where we seek feedback on the conducted exercise. In the survey, we ask the following questions: (i) "*Which program are you enrolled in?*", and (ii) "*Did the workshop help you to learn about white-box testing?*". Answer to the first question is obtained using a textbox. The second question's answer is a five-item Likert scale with five options: 'Very helpful', 'Helpful', 'Somewhat helpful', 'Little helpful', and 'Not at all helpful'.

4 RESULTS

The exercise is conducted at AU for a course titled 'Software Quality Assurance' in Fall 2022 and Spring 2023. In all, we collect responses from 124 students of which 97 and 27 are respectively, undergraduate and graduate students. Figure 1 showcases the proportion of students who perceive the authentic learning-based exercise to be helpful. From Figure 1a, we observe 77% of the surveyed undergraduate students to perceive the authentic learning-based exercise to be 'helpful' or 'very helpful'. From Figure 1b, we observe 74% of the surveyed graduate students to perceive the authentic learning-based exercise to be 'helpful' or 'very helpful'. Based on obtained data, we conclude that authentic learning is perceived positively by students to learn about white-box testing.

ACKNOWLEDGMENTS

This research was partially funded by the U.S. National Science Foundation (NSF) Award # 2247141, # 2310179, # 2209637, # 1663350, # 2312321, # 2209638, and # 2100115.

REFERENCES

[1] Paul Ammann and Jeff Offutt. 2016. *Introduction to software testing*. Cambridge University Press.

[2] Pedro Henrique Dias Valle, Armando Maciel Toda, Ellen Francine Barbosa, and José Carlos Maldonado. 2017. Educational games: A contribution to software testing education. In *2017 IEEE Frontiers in Education Conference (FIE)*. 1–8. https://doi.org/10.1109/FIE.2017.8190470

[3] Marilyn M Lombardi and Diana G Oblinger. 2007. Authentic learning for the 21st century: An overview. *Educause learning initiative* 1, 2007 (2007), 1–12.

[4] Faith W Maina. 2004. Authentic learning: Perspectives from contemporary educators. (2004).

[5] Akond Rahman, Rezvan Mahdavi-Hezaveh, and Laurie Williams. 2018. A systematic mapping study of infrastructure as code research. *Information and Software Technology* (2018). https://doi.org/10.1016/j.infsof.2018.12.004

[6] Akond Rahman, Shazibul Islam Shamim, Hossain Shahriar, and Fan Wu. 2022. Can We Use Authentic Learning to Educate Students about Secure Infrastructure as Code Development?. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 2* (Dublin, Ireland) *(ITiCSE '22)*. Association for Computing Machinery, New York, NY, USA, 631. https://doi.org/10.1145/3502717.3532125

Analyzing State-Level High School CS Teacher Certification Through Dataset Exploration

Sujeeth Goud Ramagoni
Marquette University
Milwaukee, WI, USA
sujeethgoud.ramagoni@marquette.edu

Dennis Brylow
Marquette University
Milwaukee, WI, USA
dennis.brylow@marquette.edu

ABSTRACT

Understanding the nuances of Computer Science (CS) education at the state level in high schools requires a thorough examination of complicated datasets. Numerous researchers have gathered a variety of data sources, including surveys, professional development (PD) workshops, publicly available records, enrollment statistics, and much more to gain insight into high school CS education. While these sources provide valuable starting points, the identification of CS teacher certifications remains a complex challenge. We introduce an innovative approach for the collection and analysis of confidential and public data reported to the Wisconsin Department of Public Instruction (DPI), specifically focusing on the CS teacher certification process from the 2016 to 2021 academic years (AY). Data collection related to CS education is relatively new to Wisconsin (WI), making the task of interpreting this data both challenging and intricate. Our findings highlight a net decrease in certified CS teachers in WI over the five years of available data. In this poster, we provide detailed insights into the state's CS teacher certification process and identify the current ecosystem of certified CS teachers in WI. Ultimately, every student deserves the expertise of a highly qualified teacher to guide them through CS courses.

CCS CONCEPTS

• **Applied computing** → **Education**; • **Social and professional topics** → **K-12 education**.

KEYWORDS

CS Teacher Certification, CS Certification Pathways, CS Education System Analysis, Locality

ACM Reference Format:
Sujeeth Goud Ramagoni and Dennis Brylow. 2024. Analyzing State-Level High School CS Teacher Certification Through Dataset Exploration. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635627

1 INTRODUCTION

For more than 30 years, WI has required that teachers hold a CS certification (1405 license) [13] to teach CS courses in public high schools. A CS certification for a teacher signifies that the individual has mastered the content and pedagogical standards specific to CS [12].

The Computer Science Teachers Association (CSTA) has examined the issue of CS teacher certification in the United States and found it to vary greatly across each state, with some states not requiring teachers to hold a certification in CS at all [2]. Many states have struggled to develop rigorous standards and expectations for teacher certification [7].

Our research is conducted within the framework of a statewide initiative with the overarching goal of understanding these underutilized data sources related to CS certification in WI. DPI is keenly interested in learning about the methodologies used during the data cleaning process, assessing the availability of certified CS teachers, and promoting pathways for CS certification to new teachers. This poster introduces a novel approach to identifying key aspects of the ecosystem of certified CS teachers in public high schools within these diverse and unlinked data sources, offering a holistic perspective on certified CS teacher capacity within our state.

2 RELATED WORK

A CSTA report found that the teacher certification was generally inconsistent and flawed across the U.S. [7]. The report revealed that teachers did not have adequate pathways to be certified to teach across the 50 states. A more recent report from the Expanding Computing Education Pathways (ECEP) Alliance [3] found that 12 states and the District of Columbia have approved pre-service CS teacher preparation programs and 30 states offer CS teacher certification programs [11]. To address the challenge of preparing pre-service CS teachers, a recently published book shares the various pedagogical strategies and policies that can facilitate changes in teacher knowledge and practices to teach CS concepts [8].

The ECEP Alliance Landscape Reports focus on the C.A.P.E framework, addressing capacity, access, participation, and experience in CS education [5]. While some state-level efforts have been made to address teacher capacity, a significant issue emerged in 1979 during a SIGCSE panel discussion. At that time, the panel discussed new certification standards in Wisconsin and addressed that there are only five certified CS teachers in the state, while 97 school districts reported offering CS courses [6]. A recent research study conducted by Flatland et al. highlights the necessity for certified CS teachers. To address this challenge, they primarily concentrated on in-service teachers from other disciplines, guiding them to acquire the knowledge and skills needed to teach CS [4]. Another approach involved examining the impact of student enrollment in CS courses [9] and AP CS A&P exam scores [10] based on the presence or absence of certified CS teachers.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635627

3 APPROACH

DPI serves as the primary organization tasked with gathering K-12 education data. We have collected a diverse range of datasets from DPI, which includes both confidential and publicly available data. We requested confidential data to acquire information related to the history of teachers in WI who have ever held a CS certificate.

We also used publicly available data by web scraping the DPI All-Staff Report[1] and the National Center for Education Statistics (NCES) [1]. The All-Staff Report is a published document available on the DPI website, containing information about active teachers teaching at all grade levels in WI for each AY. The CS certification data and the All-Staff Report share a primary key known as the *Entity Id*, which we used to identify the active certified CS teachers in WI. Similarly, there is a shared foreign key called the *School Code* from the NCES and certification data. This key allows us to identify the "public" high schools in WI.

The following questions will frame our exploration into CS certification data in WI:

RQ1. How are the various pathways for teachers to earn the CS certification functioning in WI?
RQ2. How are the active certified CS teachers distributed across WI based on locality?
RQ3. How many teachers are entering or departing from the system each AY?

4 PRELIMINARY RESULTS

AY	City	Town	Rural	Suburb	Total	Net Δ
2016-17	71	69	111	108	359	
2017-18	66	63	111	112	352	-1.9%
2018-19	66	62	93	109	330	-6.3%
2019-20	54	60	91	104	309	-6.4%
2020-21	52	52	91	102	297	-3.9%
Net Δ	-26.8%	-24.6%	-18%	-5.6%	-17.3%	

Table 1: Count of Active Public High School certified CS teachers for each AY based on locality. More than 17% of the certified CS teachers have departed from the system in 5 academic years.

To provide context for *RQ2* and *RQ3*, Table 1 shows a net decrease of 12 certified CS teachers from 2019-20 AY to 2020-21 AY. Between those two academic years, we observed that 33 certified CS teachers departed the system – a loss of nearly 11% of the total certified CS teachers in WI in one year. These are teachers who were active during the 2019-20 AY and were no longer reported as teaching at any high school in the 2020-21 AY.

Of the 33 departing teachers, 16 (48%) had 30+ years of experience and thus were well within the typical retirement age for public school teachers. During the same time, we identified 21 certified CS teachers who entered new high school programs in the 2020-21 AY. Out of the 21 new teachers, 12 (57%) had less than 10 years of teaching experience. Eight of the twelve teachers had 1-10 years of experience, suggesting that these teachers are coming to our system from another state. Four teachers had 0 years of experience, suggesting that these teachers are recent graduates beginning their teaching careers in CS. The remaining 9 out of 21(43%) newly entered teachers have more than 10+ years of experience, suggesting that these teachers are possibly adding on a certification in CS apart from their core subject teaching certification. This is an example of one of the four active certification pathways available for teachers in WI (*RQ1*). Other pathways for WI include: *Initial Teacher*, *School-District Sponsor*, and *Emergency Pathways*.

Table 1 also shows the percentage of net change of certified CS teachers in the state by locality and each academic year, indicating a net decrease in certified CS teachers departing the system. Interestingly, city, town, and rural areas have the highest percentage of net loss in certified CS teachers, quadrupling the loss from suburban high schools.

[1]https://publicstaffreports.dpi.wi.gov/PubStaffReport/Public/PublicReport

5 CONTRIBUTION

The preliminary analysis of state-reported CS certification data provides insight into the dynamics of how certified CS teachers enter and depart the system, contributing to the development of a model to understand WI CS teacher capacity and the functioning of certification pathways. The poster will contain visualizations showing the geographical distributions of certified CS teachers in WI. In the future, we will be collaborating with DPI and WI policymakers to bring state-level funding, allocate PD resources across different regions of WI, and implement structural changes to the pathways for receiving certification in CS.

REFERENCES

[1] National Center for Education Statistics. https://nces.ed.gov/ccd/schoolsearch/.
[2] CSTA. 2020 CSTA Standards for Computer Science Teachers. https://csteachers.org/teacherstandards.
[3] ECEP. Expanding computing education pathways (ECEP). https://ecepalliance.org/.
[4] Robin Flatland, James Matthews, Pauline White, MaryAnne Egan, and Jesse Moya. Building CS Teacher Capacity Through Comprehensive College/High School Partnerships. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, pages 606–612, 2022.
[5] Carol L Fletcher and Jayce R Warner. CAPE: A Framework for Assessing Equity Throughout the Computer Science Education Ecosystem. *Communications of the ACM*, 64(2):23–25, 2021.
[6] Timothy Fossum and Donald Chambers. A Report on Computer Science Certification for Secondary School Teachers in Wisconsin (Panel Discussion). *ACM SIGCSE Bulletin*, 11(1):245–246, 1979.
[7] Karen Lang, Ria Galanos, Joanna Goode, Deborah Seehorn, Fran Trees, Pat Phillips, and Chris Stephenson. Bugs in the System: Computer Science Teacher Certification in the US. *The Computer Science Teachers Association and The Association for Computing Machinery*, 2013.
[8] Chrystalla Mouza, Aman Yadav, and Anne Ottenbreit-Leftwich. *Preparing preservice teachers to teach computer science: Models, practices, and policies*. Research, Innovation & Methods in Educational Technology. Information Age Publishing, Incorporated, 2021.
[9] Sujeeth Goud Ramagoni and Dennis Brylow. Analyzing Certified CS Teacher Capacity and Impact on Secondary School CS Education. In *Proceedings of the ACM Conference on Global Computing Education Vol 2*, pages 201–201, 2023.
[10] Sujeeth Goud Ramagoni and Dennis Brylow. The impact of certified cs teachers on ap computer science exam scores: A study in wisconsin. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 2*, pages 30–31, 2023.
[11] Jim Stanton, Lynn Goldsmith, Richards Adrion, Sarah Dunton, Katie Hendrickson, Alan Peterfreund, Pat Yongpradit, Rebecca Zarch, and Jennifer Zinth. State of the States Landscape Report: State-level Policies Supporting Equitable K-12 Computer Science Education. *Education Development Center*, 2017.
[12] Teaching Certification. Computer Science Teacher Certification, May 2022. https://teaching-certification.com/computer-science-teacher-certification-2/.
[13] Wisconsin Department of Public Instruction. What Can I Teach with My License? https://dpi.wi.gov/licensing/general/what-can-i-teach, accessed 2023 Jan 15.

Building Awareness of Computational Thinking Pathways across a Large School District

Alessandra Rangel
Learning Experience Design
Digital Promise Global
Redwood City, CA, USA
arangel@digitalpromise.org

Merijke Coenraad
Learning Experience Design
Digital Promise Global
Washington, D.C., USA
mcoenraad@digitalpromise.org

Pati Ruiz
Learning Sciences Research
Digital Promise Global
Redwood City, CA, USA
pruiz@digitalpromise.org

Kyle Dunbar
Learning Experience Design
Digital Promise Global
Washington, D.C., USA
kdunbar@digitalpromise.org

Lisa Milenkovic
Applied Learning Department, STEM and Computer Science Program, Broward County Public Schools
Fort Lauderdale, FL, USA lisa.milenkovic@browardschools.com

Sheryl Arriola
Applied Learning Department, STEM and Computer Science Program, Broward County Public Schools
Fort Lauderdale, FL, USA
sheryl.arriola@browardschools.com

Annmargareth Marousky
Applied Learning Department, STEM and Computer Science Program, Broward County Public Schools
Fort Lauderdale, FL, USA
annmargareth.marousky@browardschools.com

ABSTRACT

As states move to adopt computer science (CS) and computational thinking (CT) standards to prepare students for an increasingly technological world, school districts are integrating curriculum with CT skills and practices, building the capacity of their teachers and administrators through professional development, and increasing the knowledge infrastructure to support CT opportunities for students. One challenge that districts face when scaling new initiatives occurs when the work becomes siloed to one "champion" teacher or a singular motivated department. This poster presents Broward County Public Schools' use of an interactive infographic to build awareness of their CT initiative, including considerations for utilizing an infographic tool and the design decisions made to engage various stakeholders. The poster provides examples of methods to build knowledge of an emerging initiative and adds to the growing body of work in supporting districts as they scale their CS/CT initiatives.

ACM Reference Format:

Alessandra Rangel, Merijke Coenraad, Pati Ruiz, Kyle Dunbar, Lisa Milenkovic, Sheryl Arriola, and Annmargareth Marousky. 2023. Building Awareness of Computational Thinking Pathways across a Large School District. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20--23, 2024, Portland, OR, USA.* https://doi.org/10.1145/3626253.3635560

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03. https://doi.org/10.1145/3626253.3635560

1 INTRODUCTION

Broward County Public Schools (BCPS) is one school district partner participating in a research practice partnership (RPP) to develop K-12 CT Pathways that are consistent, cumulative, and competency-based [1]. Through the RPP, researcher-side partners collaborated with district leaders from BCPS STEM+CS Department to begin drafting an infographic showcasing the district's CT pathway and goals. Research demonstrates that co-design is beneficial as teachers and other school experts can bring knowledge of district context, learning objectives, and needs into the design process. Additionally, co-design supports partnerships that democratize the design process and create more equitable roles for those trained in design and those who will use the outcome of the design process [2, 3].

2 OVERVIEW

In this poster, we present considerations for creating an infographic to support the integration of CT across disciplines and build administrator and teacher awareness of CT as demonstrated in BCPS. We answer the research question: *What design elements does a district highlight when creating an infographic to build awareness of their CT initiative?* The poster shares processes that district leaders identified in building the tool and has direct implications for RPPs and districts working to scale their CT initiatives. To answer the research question, we analyzed qualitative data from meeting notes with district leaders, researchers, and illustrators, survey data from a mixed-methods landscape study, and design notes across feedback cycles in the iteration of the tool.

2.1 Design Process

The idea of creating a district infographic was first considered as a way to share each district's CT Pathway and pathway goals. Following an in-person convening where BCPS district leaders presented on their CT Pathway, a recording was shared with an illustrator who then translated it into a graphic. The initial layout for the infographic included tables that cross-referenced subject areas, BCPS programs, and CT skills associated with each (Figure 1). Initial feedback from BCPS centered around whether the design told the story that they wanted. While the information was clear to the researchers and district leaders who had familiarity with CT, **the illustrator did not have background knowledge in CT and struggled to graphically display the most important information.** The result did not tell the story that the district leaders hoped for.

Despite a few meetings, the design process largely stalled until district leaders and a researcher with experience in design had an opportunity to sit down informally and sketch out some ideas in-person. The district leaders engaged with **examples of similar infographics** and collaborated to sketch visuals, provide key text, and narrow down examples that would be most relevant. While in the initial draft, BCPS identified that they wanted to highlight programs and grade levels where CT was occurring, ultimately, they decided that this was secondary to surfacing examples of CT that teachers, students, and administrators would recognize. The final mocked-up design was sent to the illustrator to be turned into a formal illustration (Figure 1). In the end, we found that it was necessary for the research team to create a rough prototype that translated district ideas and our shared conceptualizations of CT to the illustrator to ensure the infographic aligned to district needs and accurately presented CT and their pathway.

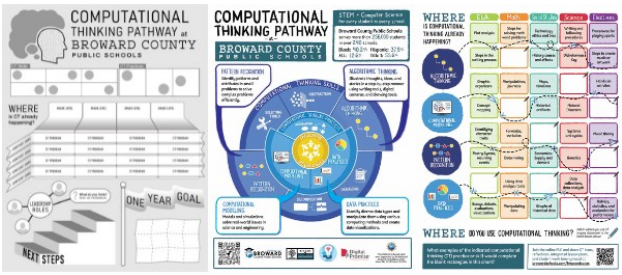


Figure 1: Progression of BCPS infographic from initial draft (left) to final version (middle, right). [Image Credit: Digital Promise]

2.2 Design Elements

Throughout the design process, district leaders identified three main design elements that needed to be incorporated into the infographic: 1) **interactivity**, 2) **CT integration strategies and examples**, and 3) **graphic representation of CT skills and practices**. The infographic was meant to be **interactive**, such that the user (teachers, school leaders, students, and partners) could engage with through QR codes or questions such as "Where do you use computational thinking?" Adding design directives, such as pencils, blank boxes, and arrows provide the user the opportunity to fill in the blanks and begin seeing connections between the CT skill depicted and their own work. To incorporate **CT integration strategies and examples**, district leaders and researchers focused on the most prevalent CT skills and practices reported by teachers in a survey, including algorithmic thinking, pattern recognition, data practices, and computational modeling. **Graphically**, the infographic adapts visuals that teachers and administrators would be familiar with based on previous work with the district CT pathway, and would provide a starting point for those unfamiliar with the work. The central concentric circle figure, drawn from prior work [4], showcases all CT skills and practices, while slightly larger callout boxes demonstrate the four focal areas and callout boxes define each skill or practice. The use of this graphic was purposeful as it allows for updates as new skills or practices are emphasized within the district.

3 FUTURE WORK

District leaders plan to continue to iterate on the infographic, updating examples as teacher knowledge deepens to highlight new CT skills and practices like abstraction and automation. Currently, district leaders have added the infographic to their department webpage and have used it across teacher PD as a discussion piece for teachers to use as a "starting document" and incorporating slides alongside the infographic that can be used across PD topics. Teachers can easily pick up the document, in printed or digital form, and share it with their colleagues and building leaders. In one year, district leaders expect the infographic will support their work in scaling CT across the district and hope to continue to see growing engagement with teachers implementing CT in their classrooms, participating in PD, and sharing the infographic, while other districts participating in the RPP can adapt the BCPS model in their own district and CT Pathway context.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2219350 & 2219351.

REFERENCES

[1] Merijke Coenraad, Quinn Burke, Pati Ruiz, Kelly Mills, and Jeremy Roschelle. 2022. Developing inclusive computing with the CT Pathways Toolkit. In SIGCSE 2022 - Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V.2, Providence, RI, USA, 1089. DOI:https://doi.org/10.1145/3478432.3499076

[2] William R. Penuel, Jeremy Roschelle, and Nicole Shechtman. 2007. Designing formative assessment software with teachers: An analysis of the co-design process. *Res. Pract. Technol. Enhanc. Learn.* 2, 1 (2007), 51–74. DOI:https://doi.org/10.1142/S1793206807000300

[3] Michael Scaife, Yvonne Rogers, Frances Aldrich, and Matt Davies. 1997. Designing for or designing with? Informant design for interactive learning environments. In *CHI '97 Proceedings of the ACM SIGCHI Conference on Human factors in computing system*, 343–350. Retrieved September 22, 2017 from http://ldt.stanford.edu/~jsulzen/james-sulzen-portfolio/classes/ED229b/Readings/ED229B-Readings/www.stanford.edu/class/ed229b/fall00/readings/p343-scaife.pdf

[4] Kelly Mills, Merijke Coenraad, Pati Ruiz, Quinn Burke, and Josh Weisgrau. 2021. Computational Thinking for an Inclusive World: A Resource for Educators to Learn and Lead. Digital Promise. DOI: https://doi.org/10.51388/20.500.12265/13

Student (Self) Assessment, Feedback, and Relatability in CS-2 Game Programming Assignments

Veena Ravishankar
vravisha@umw.edu
University of Mary Washington
Fredericksburg, Virginia, USA

Jessica Zeitz
jzeitz@umw.edu
University of Mary Washington
Fredericksburg, Virginia, USA

ABSTRACT

Research has shown incorporating game programming and relatable topics benefits student learning and increases engagement. When enrolled in our CS-2 object-oriented analysis and design course, students complete a series of individual and group projects. We seek to gauge how incorporating these large-scale game-based programming assignments based on relatable topics affect students' learning outcomes and (self) assessment of their knowledge in terms of their confidence and capabilities. Our assessment is conducted through pre- and post-surveys that include knowledge areas such as object-oriented analysis, design and testing that map to learning outcomes of the course. We additionally assess learning outcomes through demonstrations of their implemented coding projects and feedback. We present initial findings of these results.

ACM Reference Format:
Veena Ravishankar and Jessica Zeitz. 2024. Student (Self) Assessment, Feedback, and Relatability in CS-2 Game Programming Assignments. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635531

1 BACKGROUND AND CONTEXT

Object-oriented programming (OOP) is widely used in the computer science field and is included in many curriculums at the college and university level [2]. To teach OOP, many institutions have incorporated game programming [1, 3, 4]. Authors in [8] have implemented a set of relatable game programming assignments into their OOP course. Game programming has been proven beneficial in teaching OOP concepts as well as in keeping students engaged in the material. When students are engaged, they are also more successful [6, 7]. Research [5] has shown the importance of introducing and including relatable topics in teaching Computer Science early on to keep students more engaged, gender equitable and successful. The assignments are task and game based, and the topic chosen is set specific to our campus, which the students are familiar with and can relate to easily.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635531

2 OVERVIEW OF OUR ASSIGNMENTS AND ASSESSMENTS

The assignments are divided into three main parts: two individual and a group project. The individual assignments give students a foundation for the large scale group part and allow them to become stronger coders on their own. The topic of the projects is for students to design a program that tours campus and performs various activities they are involved in across campus. Students incrementally add various features they can relate to associated with our campus.

Student learning assessment was twofold. First, students demonstrated their completed projects via interactive evaluations throughout the semester. (This was in addition to the standard grading, i.e., grading their projects by testing their code and design individually.) During project demonstrations, students explained the code features they implemented based on the problem statement provided, presented different scenarios of touring our campus virtually and went through different use cases of the problem design. We evaluated their performance based on the accomplishment of project features that directly map to learning outcomes. Second, we introduced a pre and post survey to measure knowledge confidence and topic familiarity via self-assessment. We included questions in the survey to measure their confidence in Object Oriented design and programming concepts, their comfort with working in groups, and their experience working on large-scale Java projects of different types and themes.

We evaluated 35 students over 3 semesters from our institution's CS-2 course, starting in Fall 2022. The groups were self-assigned by students during the first and last semesters of this study, and pre-assigned by the instructor during the second based on their performance on early assignments. We assigned a pre-survey in the beginning of the semester and collected post-survey results at the end of the semester.

3 PRELIMINARY RESULTS

Out of the 35 students, we had a total of 22 respondents who took both the pre- and post-surveys from all three semesters combined. While the pool is small, this size is typical of our small liberal arts setting. Our classes had an enrollment of about 10 students per semester. Both of the assessment techniques deepened our understanding of students' knowledge of the core concepts taught in class and gave us detailed feedback directly from the students. We report initial results on student self-assessment about performance, group work experience and their motivation based on different themes for the projects.

On both surveys, we asked students to rate their confidence in eight knowledge areas on a scale from 1 to 5. The knowledge areas

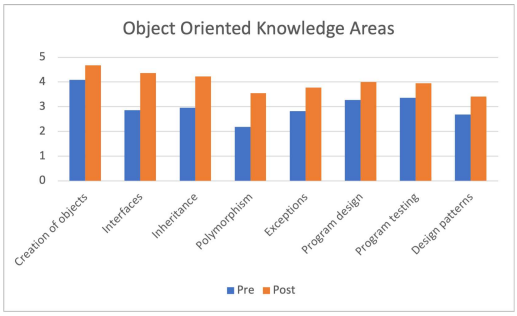


Figure 1: Students' confidence in their knowledge of object oriented concepts

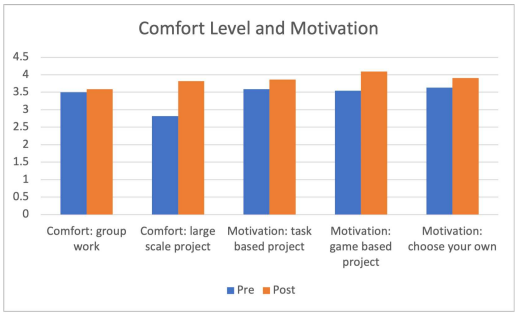


Figure 2: Average student comfort level with group work and average student motivation based on project type

are all main topics in our CS-2 course. In Figure 1, we report students' self-assessment of their confidence on these object-oriented concepts at the beginning and the end of the course. Clearly, we see an improvement in their confidence over all topics.

We asked students to rate their comfort level when working in groups in general and working on a large-scale project both on a scale from extremely uncomfortable to extremely comfortable. We also asked students to rate their motivation level when working on different types of projects. In Figure 2, students' self-assessment of their motivation and comfort levels increased slightly after completing the course. This suggests that their confidence and enthusiasm in these areas have grown as a result of their coursework. Over half of the students reported positively when specifically asked to describe their group project experience in this CS-2 course.

This course includes two individual projects, one group project and weekly labs. All learning outcomes of this course directly map to one or more of the projects as seen in Table 1. Throughout the semester, students interactively demonstrated their projects to us.

During the demo of individual projects, students presented their project features. This gave us an understanding of how well they learned the Object Oriented design and programming concepts, and helped us gauge their confidence and ease with the project theme. Based on the student experiences and feedback, changes or updates were made to aid student success and help with the progress throughout this course offering. Scaffolding was added to individual Project 2 based on the reception of Project 1, and detailed steps added accordingly. Weekly labs were simultaneously updated

Table 1: Mapping Student Learning Outcomes to Projects

SLOs	Associated Project & Features
1. Recognize and explain the role of encapsulation, visibility, inheritance and polymorphism in object oriented design	Individual projects 1 & 2
2. Identify and document requirements using Use Cases	Group project
3. Use design techniques to identify classes and methods and capture them in UML class diagrams	Group project
4. Identify test cases and apply them to a project	Group project
5. Develop larger scale programs in a collaborative, team environment	Group project
6. Recognize design patterns and the application of common patterns	All three projects
7. Recognize diversity and inclusion issues and their effects within the field	Students incorporated concepts into group project design features

to reinforce these Object Oriented and design concepts. During the group project demos, students presented their chosen project features which mirrored familiar activities performed across campus and reflected their inclusivity and equity values. These features varied from interactions with the campus community, displaying energy bars and our university mascot, and scores assigned based on participation in diverse club activities, to mention a few.

The students demonstrated enhanced performance and learning in each successive project demo, reflecting the positive impact of integrating their feedback and implementing changes or additions to the project. This iterative process of feedback and improvement contributed significantly to their success, growth and development.

4 OUTCOMES AND FUTURE WORK

As we hypothesized, students were more confident in their knowledge after completing the projects in this course. They were also more comfortable working in groups on a large-scale programming project. The theme of the game-based projects was related to our institution to increase relatability. In future research, we aim to explore the influence of theme on the retention of underrepresented populations by gathering more comprehensive data.

REFERENCES

[1] Matt Barton and Shane Stacks. 2019. *Dungeons and desktops: The history of computer role-playing games 2e.* CRC Press.
[2] Jessica D Bayliss. 2009. Using games in introductory courses: tips from the trenches. In *Proceedings of the 40th ACM technical symposium on Computer science education.* 337–341.
[3] Woei-Kae Chen and Yu Chin Cheng. 2007. Teaching object-oriented programming laboratory with computer game programming. *IEEE Transactions on Education* 50, 3 (2007), 197–203.
[4] Robin M Givens. 2020. A scalable RPG project for object-oriented software development. *Journal of Computing Sciences in Colleges* 36, 3 (2020), 53–62.
[5] Kai Marquardt, Ingo Wagner, and Lucia Happe. 2023. Engaging girls in computer science: Do single-gender interdisciplinary classes help?. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET).* IEEE, 128–140.
[6] Mike Wallinga. 2019. Engaging CS2 students via a semester-long in-class game project. *Journal of Computing Sciences in Colleges* 35, 2 (2019), 77–88.
[7] Lu Yan. 2009. Teaching object-oriented programming with games. In *2009 Sixth International Conference on Information Technology: New Generations.* IEEE, 969–974.
[8] Jessica Zeitz and Veena Ravishankar. 2021. Text-based large-scale programming assignment for teaching object-oriented analysis and design: faculty poster abstract. *Journal of Computing Sciences in Colleges* 37, 3 (2021), 157–157.

Experience Helps, but It Isn't Everything: Exploring Causes of Affective State in Novice Programmers

Heidi Reichert
hreiche@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Sandeep Sthapit
ssthapi@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Benyamin T. Tabarsi
btaghiz@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Ally Limke
anlimke@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Thomas Price
twprice@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Tiffany Barnes
tmbarnes@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

ABSTRACT

Affective state, referring to an individual's feeling, can impact students' confidence and retention in CS, particularly for novice programmers. However, little research has been conducted to examine how moments that occur during programming impact students' affective states in real-time. In this pilot study, seven undergraduate students in an introductory block-based programming course completed a programming assignment and were surveyed and interviewed about their experience and self-efficacy as programmers. While programming, students periodically recorded their affective states via a popup in the programming environment. We performed retrospective think-aloud interviews with students afterward, asking them to watch and reflect on recordings of their programming. We subsequently analyzed student interviews using thematic analysis to derive 206 codes. These codes were grouped into three areas that impacted affect: the environment, objective progress, and perceptions during programming. To explore why students responded as they did to moment occurrence, we further categorized students based on four dimensions: programming experience, assignment completion, confidence, and the impact of the programming session on self-efficacy. Our initial results suggest that while certain moments elicit similar affective states among students, the interaction of the aforementioned four dimensions may have a higher impact on novices' affective states during programming. We conclude with recommendations for educators to improve students' affective states during and after programming.

ACM Reference Format:
Heidi Reichert, Sandeep Sthapit, Benyamin T. Tabarsi, Ally Limke, Thomas Price, and Tiffany Barnes. 2024. Experience Helps, but It Isn't Everything: Exploring Causes of Affective State in Novice Programmers. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635508

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635508

1 INTRODUCTION AND RELATED WORK

There is a well-known need to have more and more diverse students enter the professional world of computer science (CS), which is often addressed in CS education through interventions to improve performance, sense of belonging, or self-efficacy. In this context, it makes sense to consider how CS education might best promote positive experiences and reduce negative experiences while novices learn to program. But what experiences actually impact novice programming students during programming?

Linnenbrink [2] defined the affective state as a situation-oriented reaction to the changing environment. She also pointed out that affective states bi-directionally link to cognition and motivation. Students' affective states and emotions play an important role in their learning and performance [1**?**]. Since programming courses are challenging [5], this role is even more significant.

The impact of programming experiences on novice student affect, particularly in block-based programming (BBP) environments, remains relatively understudied. An understanding of how positive and negative affective moments impact self-efficacy for novice programmers is critical in understanding how to improve students' confidence and retention [6].

2 APPROACH

We designed a tool that provided two ways to report affect. Students could hover over the button on the bottom right corner of the BBP interface, which featured the words "How are you feeling right now?" When hovered over, the button expanded into a box with nine affect options based on a study by Sabourin et al. [4], with modifications to allow students to report positive emotions and multiple emotions. Researchers could also trigger the reporting tool in a Wizard of Oz fashion, which required the students to pause programming and report their emotions to the tool.

Undergraduates in a CS0 course for non-computing majors were invited to remotely participate in this study over Zoom. Seven participants signed up for a 90-minute time slot, which consisted of a programming task in the BBP language Snap*!* (30 min), a survey, a short open-ended interview, and finally, a retrospective interview (30 min). Prior to programming, students were introduced to the aforementioned affect tool and informed that a request to report their emotions would periodically pop up. During the programming session, the researcher monitored the student remotely. At

key moments, the tool was prompted by the researcher on the participant's screen. These moments were identified by the researcher, who based them on significant events, such as when the student ran their code, or behavioral cues, like moments of reflection. This occurred at regular intervals, at least once every 5 minutes.

Following the programming session, students completed a brief survey about their confidence as programmers and their programming experience. They were then asked a few open-ended questions about their programming experience to understand how they felt about the programming session as a whole rather than at specific moments, and how the session impacted their self-perceptions around programming. Students then watched a recording of their session with the researcher. During the retrospective viewing, students were asked to share how they felt at each moment of the session. We specifically asked students to discuss the moments in which they reported an affective state using the tool and how the affective state related to their views of themselves as programmers. To investigate the moments which significantly impacted student affect and self-assessment, we transcribed the interviews of all seven students. Afterward, we performed a thematic analysis using open coding on the transcripts [3]. Whenever a student shared they were experiencing a specific emotion, we tagged the emotion and also a description of the programming environment/moment.

3 RESEARCH CONTRIBUTIONS

In this work, we developed new methods for data collection, as well as provided categorizations using a grounded approach that may help understand the affective state of novice programmers.

First, we developed a new tool to measure affective state within the Snap*!* environment and prompt moments that provided foundations for future discussions via interview. This allows students to both "vent" their emotions while programming, as well as researchers or instructors, to gather real-time data on how students feel while programming. **Second**, to our knowledge, this is some of the first work to utilize the retrospective interview approach, in which students are asked to comment and reflect on a recording of their programming experience almost immediately post-programming. This approach allowed for a deeper understanding of students' thoughts while programming, during which the interviewer could ask questions and prompt discussion during moments of interest. **Third**, we proposed some factors that may explain why students were impacted differently by the same moments. While tagging and coding for affective states and moments that occurred during programming, we discovered significant differences between students. Encountering errors, for instance, inspired both feelings of curiosity and frustration. Other moments, like having to stop to plan during programming, featured similar differences in perception among students. While reviewing students' transcripts, survey responses, and programming sessions, we determined that there were four primary traits that separated students' affective states. These were prior programming experience, assignment completion, confidence post-programming, and the impact of the programming session on the student's sense of self-efficacy.

We found that all students who successfully completed their programs had high post-programming confidence. However, this completion was not a prerequisite for confidence. Students who enter programming classes with less experience may be more prone to placing importance on their programming experiences and are consequently a demographic that requires more attention for early intervention efforts that target improving affect and self-efficacy. Successful programming experiences help to support confidence and an improved sense of self as a programmer; however, this ideal is not always reached by students. In examining the data from students we interviewed who had less experience, less confidence, and an inability to complete the assignment, the distinguishing factor we found was self-comparison – a student acknowledging how far they had come, in comparison with themselves at a previous point in time, improved their post-programming affect.

4 CONCLUSIONS AND FUTURE WORK

While further research is necessary to corroborate our findings, our preliminary results suggest that instructors' promotion of explicit reflections on improvement over the course of an introductory exercise or class could improve students' affect. It may also be the case that providing explicit information to students about what kinds of difficulties and challenges are normal can impact their self-assessment. Finally, several students during this study appreciated having a button to express their emotions during programming. Further incorporation of these types of affective state tools into IDEs may give willing students the opportunity to communicate how they feel during their programming experiences and allow them to feel heard. However, this work is limited by its size, as only seven participants made up this study. The number of affective states we collected was limited by this population and our single programming session, and it is potentially the case that more cases exist for students. This study also only administered a post-programming survey. As a result, data regarding students' initial confidence levels before entering the study and working on the assignments are missing. Further study and replication of our early findings are warranted.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant number 1917885.

REFERENCES

[1] Rafael A. Calvo and Sidney D'Mello. 2010. Affect Detection: An Interdisciplinary Review of Models, Methods, and Their Applications. *IEEE Transactions on Affective Computing* 1, 1 (Jan. 2010), 18–37. https://doi.org/10.1109/T-AFFC.2010.1 Conference Name: IEEE Transactions on Affective Computing.

[2] Elizabeth A. Linnenbrink. 2006. Emotion research in education: Theoretical and methodological perspectives on the integration of affect, motivation, and cognition. *Educational Psychology Review* 18 (2006), 307–314. https://doi.org/10.1007/s10648-006-9028-x Place: Germany Publisher: Springer.

[3] Moira Maguire and Brid Delahunt. 2017. Doing a thematic analysis: A practical, step-by-step guide for learning and teaching scholars. *All Ireland Journal of Higher Education* 9, 3 (2017).

[4] Jennifer Sabourin and James Lester. 2014. Affect and Engagement in Game-Based Learning Environments. *Affective Computing, IEEE Transactions on* 5 (01 2014), 45–56. https://doi.org/10.1109/T-AFFC.2013.27

[5] Benyamin T. Tabarsi, Ally Limke, Heidi Reichert, Rachel Qualls, Thomas Price, Chris Martens, and Tiffany Barnes. 2022. How to Catch Novice Programmers' Struggles: Detecting Moments of Struggle in Open-Ended Block-Based Programming Projects using Trace Log Data. Zenodo. https://doi.org/10.5281/zenodo.6983260

[6] Mylène Ward, Larry Gruppen, and Glenn Regehr. 2002. Measuring Self-assessment: Current State of the Art. *Advances in Health Sciences Education* 7, 1 (Jan. 2002), 63–80. https://doi.org/10.1023/A:1014585522084

Using Concept Maps for Notional Machine Selection in CS1

Ethan Richards
Colorado School of Mines
Golden, Colorado, U.S.A.
erichards@mines.edu

Sonia Spindt
Cherry Creek Innovation Campus
Centennial, Colorado, U.S.A.
sspindt@cherrycreekschools.org

Gabriel Fierro
Colorado School of Mines
Golden, Colorado, U.S.A.
gtfierro@mines.edu

ABSTRACT

A notional machine is an abstract representation of a program's execution. These alternate representations serve to highlight otherwise hidden or subtle behavior in program execution, and eliminate unnecessary or irrelevant details that can obstruct understanding. Prior work has established that, by construction, there is no "universal" notional machine that is appropriate for the entirety of a CS1 curriculum. How, then, does an educator choose an appropriate notional machine for a given learning sequence?

We present initial work on helping educators choose an effective sequence of notional machines for a CS1 course. Our key insight is to leverage a concept map capturing hard and soft dependencies between computing concepts. By characterizing concepts and notional machines together, we show that it is possible to design feasible sequences of notional machines that cover a desired learning sequence.

ACM Reference Format:
Ethan Richards, Sonia Spindt, and Gabriel Fierro. 2024. Using Concept Maps for Notional Machine Selection in CS1. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635535

1 INTRODUCTION

Notional machines (NMs) are a tool for explaining computing concepts in different or abstracted terms. These alternate representations serve to highlight otherwise hidden or subtle behavior in program execution, and eliminate unnecessary or irrelevant details that can obstruct understanding. Prior work has established that, by construction, there is no "universal" notional machine that is appropriate for the entirety of a CS1 curriculum [2, 4]. Thus, the natural question arises: how does an educator choose an appropriate notional machine for a given learning sequence?

Our key insight is to root this decision making process in a CS1 concept map that captures hard and soft dependencies between computing concepts. We represent the map as a directed graph so a learning sequence can be described as a traversal of that graph. To inform the choice of notional machines for a given learning sequence, we look to recent work in characterizing notional machines by their representation and concepts covered [1]. Through annotating the notional machines with the same concepts as captured in the map, we show that it is possible to design and evaluate different sequences of notional machines that complement the desired learning sequence.

In this poster abstract, we show our initial work in using a hypothetical CS1 concept map to choose a sequence of notional machines given a learning sequence over the concept map.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635535

2 BACKGROUND AND RELATED WORK

A notional machine is an abstract representation of a program's execution [4]. Fincher [1] proposes two forms in which this representation can manifest: a representation-based interactive activity or an analogy-based representation. Such representations are beneficial to a learner to understand topics and form their own mental models of program flow and state.

[2] asserts that here is no single "universal" notional machine; that is, there is no single abstraction sufficient to cover all CS1 concepts. This means that a full learning sequence for a CS1 course would require multiple notional machines, whether analogy-based, representation-based, or otherwise. However, little work has been done in mapping sequences of notional machines to curricula.

Concept maps are an established idea in pedagogy [3]. Such maps can be used to better inform the development and design of curriculum by portraying it in a sequence, and then adapting material based on the constraints of the educator.

However, such sequences are not guaranteed to be linear, and have no such description of dependence between concepts. As a result, there is a need for helping educators choose an effective sequence of notional machines. We propose that if we express concept maps and notional machines together using a directed graph, we can leverage the graph's structure to inform the selection of sequences of notional machines in a CS1 curriculum.

3 CONCEPT MAP FOR CS1

Educators have inherent constraints such as time, domain knowledge, student interest, or mandated curriculum (e.g., AP). Concept maps can help to make obvious the distinctions within a directed set of concepts and how to best approach designing a learning sequence given those constraints.

Figure 1 shows a simplified concept map with multiple possible learning sequences for teaching functions. Each node represents a single programming concept, such as variables. A programming concept is a generalized concept such that it can relate to other concepts, and can have child sub-concepts of its own. For simplicity and clarity of presentation, we illustrate only a subset of our larger concept map that we have developed. Notional machines in the figure are drawn from [1].

3.1 Relationships

In our formulation, concept maps have two types of relationships: hard prerequisites and soft prerequisites. **Hard prerequisites** capture which concepts *must* be covered before another concept can be taught. For example, functions are a natural hard prerequisite of teaching recursion.

Soft prerequisites capture optional, but helpful, dependencies between concepts. For example, while we contend that an understanding of data types in programming languages is essential knowledge for understanding variables as containers of state (hard prerequisite), a conceptualization of expressions is not strictly necessary. This does limit the kinds of programs that can be expressed by the learner (i.e. only those with static assignments to variables). We capture this concept topology in Figure 1.

3.2 Learning Sequences

The intent of the concept map is to capture all of the *hard* and *soft* dependencies between programming concepts. The process of designing a learning sequence can thus be framed as choosing a topological sort of this graph such that (a) hard prerequisites must be covered before their dependent concepts, and (b) soft prerequisites *may* be covered before their dependent concepts. The concept map offers a principled way of choosing which concepts to teach, and in what order. Further, we will show how the concept map can inform the choice of notional machines (§4).

We first illustrate how our (intentionally) simple concept map permits multiple learning sequences for starting from data types and progressing to teaching functions. The first sequence (`a-c-e`) minimizes the number of concepts covered before functions are taught by ignoring any soft prerequisites. Another choice (`a-b-c-e`) includes expressions in the learning sequence to expand the set of programs that learners can write; here, expressions are taught before variables because of the soft prerequisite relationship.

Neither of these learning sequences requires teaching control flow. However, if the educator decides to include recursion as a learning goal, then the concept map will lead them to include control flow as a necessary component of the learning sequence. The topology of the map allows them to cover control flow either before or after functions, as long as it is covered before teaching recursion. This also changes the potential sequence of notional machines.

Ultimately, there is no one correct way of mapping notional machines to concepts in any given curriculum, but we intend to give a set of options to educators that are useful enough to produce a learning sequence.

4 CHOOSING A NOTIONAL MACHINE SEQUENCE

Notional machines can be characterized in terms of program concepts using a concept map. This characterization can be applied with a graph-theoretic approach in order to choose a sequence of notional machines to use in a curriculum.

Using the concept map, the educator can determine which notional machines can be used to teach or illustrate a given concept. With the added context of dependencies between concepts, the educator can also determine how a particular *sequence* of notional machines can best support the learner experience.

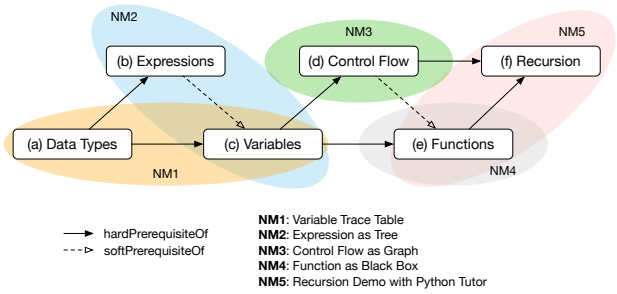


Figure 1: A simple concept map allows multiple possible learning sequences for teaching Functions: `a-c-e`, `a-b-c-e` and `a-c-d-e`. Notional Machines (NM*X*) are drawn from [1]

Consider the sequence `a-c-d-e` from Figure 1. This sequence can be covered with notional machines NM1, NM3 and NM4. NM2 would not necessarily be appropriate to use for teaching variables because its design also incorporates expressions, which are not in the learning sequence. The use of the concept map and notional machine characterization supports this choice. If an educator wished to extend the above sequence to also include recursion (`a-c-d-e-f`), then they must make a decision whether to (a) use NM4 to teach functions and then NM5 to teach recursion, or (b) use NM5 to teach both functions and recursion.

One caveat within a sequence is that transitioning between notional machines may be confusing for learners. For that reason, an educator might want to minimize the number of notional machines used to cover concepts in a sequence. Expressing a learning sequence with concepts and notional machines together is an effective tool to solve this issue; educators can reason about when the transition between one notional machine to another may or may not be appropriate and timely for students.

Our proposed approach enables informed co-design of learning sequences and notional machines. However, it is ultimately the responsibility of the educator to decide which sequences of notional machines are beneficial to implement in their learning sequence.

5 CONCLUSION

Graph theory can be utilized with concept maps to better inform our decisions about the sequence of notional machines in CS1 curricula. Future work will approach the problem of determining "good" sequences of concepts and notional machines algorithmically.

REFERENCES

[1] Sally Fincher, Johan Jeuring, Craig S Miller, Peter Donaldson, Benedict Du Boulay, Matthias Hauswirth, Arto Hellas, Felienne Hermans, Colleen Lewis, Andreas Mühling, et al. 2020. Notional machines in computing education: The education of attention. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*. 21–50.

[2] Sally A Fincher and Anthony V Robins. 2019. *The Cambridge handbook of computing education research*. Cambridge University Press.

[3] Vinicius dos SANTOS, Érica F de SOUZA, Katia R Felizardo, and Nandamudi L Vijaykumar. 2017. Analyzing the use of concept maps in computer science: A systematic mapping study. *Informatics in Education* 16, 2 (2017), 257–288.

[4] J Sorva. 2013. Notional Machines and Introductory Programming Education. *ACM Transactions on Computing Education (TOCE)* 13, 2 (2013), 8–1.

Becoming Core: Curriculum Planning Tools for Integrating CS into K-5 Content Areas

Darcy M. Ronan
ronand@sacredheart.edu
College of Education & Human Development, Sacred Heart University
Fairfield, CT, USA

Heidi Williams
heidi.williams@marquette.edu
Marquette University
Milwaukee, WI, USA

ABSTRACT

In the K-5 setting, introduction of computer science (CS) through integration with other content areas is a promising pathway to expand CS access in a manner which is meaningful, accessible, sustainable, and scalable. We have developed a suite of planning tools to support CS-novice elementary classroom teachers (N = 44) to develop CS-embedded units through iterative planning and feedback. A unit template, grounded in best practices like Universal Design for Learning and Understanding by Design, progresses through foci on *CS Connections, Unpacking, Instructional Resources, Assessment,* and *Supports & Meaning* to teach CS and another area side-by-side. A matched set of checklists prompt self-reflection while rubrics define high-quality attributes for use in self, peer, or expert feedback. Our experience using these tools has been positive; 100% of survey respondents agreed the tools were successful in supporting the design process (N = 19). Our experience with these tools has shown a proof-of-concept for elementary classroom teachers as designers of CS-embedded units. Grounded in and responsive to teachers' real-world target content area curriculum expectations, these units offer a path to institutionalizing CS.

CCS CONCEPTS

• **Applied computing → Education**; • **Social and professional topics → K-12 education**; **Model curricula**.

KEYWORDS

CS Integration, Teacher Professional Learning, Elementary School, Primary School, CS Curriculum Co-Design

ACM Reference Format:
Darcy M. Ronan and Heidi Williams. 2024. Becoming Core: Curriculum Planning Tools for Integrating CS into K-5 Content Areas. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635588

1 INTRODUCTION

Early adopters, champions, and innovators in Computer Science (CS) education have made impressive progress in promoting the accessibility of CS to young learners. The need for early engagement with CS has been supported through learning progressions [5], standards [4], curriculum development [2], and professional learning [7].Yet gaps in capacity and access persist [1]across the K-12 spectrum. While presenting challenges for CS, elementary schools offer much richness and possibility as sites for research and innovation.

Foremost among the tantalizing organizational affordances of elementary school is the typical self-contained classroom with its built-in potential for CS curriculum integration. The connections and applications of CS with other content areas have been explored to promising effects in various grade bands [6], [15] with the potential to drive both CS learning outcomes and the target content area. Yet, many administrators consider it a dilemma to "make room" for CS, STEM, and/or science in a crowded, high-stakes, K-5 curriculum. Classroom generalist teachers also face challenges as teachers of CS. They have many curriculum areas to attend to and may have limited background, interest, and self-efficacy in CS [8]. By surfacing natural connections between CS and other areas classroom generalists are already responsible for, integrated CS becomes more accessible for teachers and more meaningful for all. Integration is a strategy for acceptance.

Elementary CS integration is a burgeoning topic, with new digital resources supporting teachers adding CS lessons to existing units in the target content area, such as through Code.org's CS Connections [3]. Our tools attend the challenges of building integrated units to teach CS topics alongside another content area. Consensus best practices for curriculum development in our context include standards alignment [4], [10], backwards design [14], Universal Design for Learning [11], and culturally sustaining pedagogy [12], [9]. Curriculum tools for CS-embedded units should reflect these principles - both to support high-quality design and to support uptake of CS into established systems.

2 PROJECT CONTEXT

We designed this suite of tools to support teachers in a large, grant-funded project to develop and implement CS-embedded units in grades K-5, Project {FUTURE}. The project provided introductory CS professional learning for partner schools (N = 28) through Code.org CS Fundamentals [2] workshops for K-5 faculty (N = 798), with follow-on support provided through a coaching program. A smaller group of teachers (N = 44) then opted into the unit development Collaborative. This two year professional learning program included a five day institute each summer as well as quarterly check-ins and workshops during the academic year. The unit tools were introduced in the first week-long summer institute. Teachers self-formed into Professional Learning Communities

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635588

(PLCs) based on grade-levels and content topics of interest. While they were coached to consider locally defined high priority learning outcomes, teachers were given the freedom to explore and select any CS integration that would be would be relevant and applicable to them. A total of 17 units are being developed by 44 teachers in two states.

3 DESIGN

Following in the tradition of Understanding by Design (UbD) [14], we wanted the curriculum planning tools to guide teachers through a series of foci to support quality design. We defined phases for curriculum planning as described below.

In the *Making Connections* phase, we begin with the core insight required to enter into the unit planning process: a logical connection between CS and a target content area. Next teachers move into the *Unpacking* phase, with its categories for vocabulary, concepts, skills, practices, objectives, etc. Next, teachers enter the third focus, *Unit Context & Instructional Resources*. In this section, teachers survey relevant existing resources in both CS and the target content area with an eye towards remixing and adapting materials. Next teachers progress to the *Assessment* design phase with its focus on conceptualizing the culminating student product or performance. The final focus area is *Supports & Meaning*. Teachers check and refine their unit plans through the lenses of accessibility and equity, using techniques of differentiation [13]and culturally-sustaining pedagogy [12] under the framework of Universal Design for Learning [11].

4 CURRICULUM TOOLS

To support these design phases, we created a suite of planning tools. These include a unit template, checklists for each phase, and rubrics for each phase. We color-coded the phases and used language and structure consistently to provide an aesthetically cohesive experience. The full set of documents is available for use at the Project {FUTURE} website (www.projectfuturecs.org).

4.1 Unit Template

It was important to balance an overall sequential progression through the phases with iteration across them. To promote this balance, we created a *Unit Template* which centers each phase in sequence, while prompting teachers to look ahead and reflect back. To guide the development process, prompting text is present in each field, which may be a guiding question or a direction.

4.2 Checklists & Rubrics

As teachers completed each phase of the template, they were invited to review the matched *checklist* as a reflection. Each phase has a corresponding *rubric* which communicates expectations and aspirations for quality [10] through 3-4 key criteria in each domain. These quality descriptors attend to the specific requirements of ensuring integrated units deliver quality CS learning alongside the target content area.

5 IMPACTS & IMPLICATIONS

Each PLC submitted their work-in-progress for review in each phase. In the program evaluation survey, summer institute attendees were asked about their experience with the tools and templates alongside other elements of the Collaborative. All teacher responses (N = 19) indicated Agree or Strongly Agree that the planning tools had been successful in supporting their unit design process.

Project {FUTURE} units include CS integration with mathematics, reading, writing, science, social studies, music, art, and social emotional learning. The teachers in Project {FUTURE} had limited experience with learning and teaching CS yet were successful in accessing CS standards, pedagogy, and curriculum through the planning tools to surface meaningful CS integration topics at the K-5 level.

ACKNOWLEDGMENTS

Project {FUTURE} is funded by the United States Department of Education's EIR Program, Award U411C190254

REFERENCES

[1] Code.org. 2022. 2022 State of Computer Science Education. https://advocacy.code.org/stateofcs
[2] Code.org. 2022. Computer Science Fundamentals. https://code.org/educate/curriculum/elementary-school
[3] Code.org. 2023. Computer Science Connections. https://code.org/educate/csc
[4] Computer Science Teachers Association. 2017. CSTA K12 Computer Science Standards. https://www.csteachers.org/page/standards
[5] Computer Science Teachers Association and the Association for Computing Machinery, Inc. 2012. Computer Science K–8: Building a Strong Foundation. https://csteachers.org/documents/en-us/25b0b43f-0817-4c08-9210-f1a96f2b1c7d/1/
[6] Kathi Fisler, Emmanuel Schanzer, Steve Weimar, Annie Fetter, K. Ann Renninger, Shriram Krishnamurthi, Joe Gibbs Politz, Benjamin Lerner, Jennifer Poole, and Christine Koerner. 2021. Evolving a K-12 Curriculum for Integrating Computer Science into Mathematics. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 59–65. https://doi.org/10.1145/3408877.3432546
[7] International Society for Technology in Education. 2018. ISTE Standards for Educators: Computational Thinking Competencies. https://www.iste.org/standards/computational-thinking
[8] Laura C. Karl. 2011. *Elementary Teachers' Perceptions of Technology Proficiencies and Motivation to Integrate Technology in School Curriculi.* Ph. D. Dissertation. Walden University. https://scholarworks.waldenu.edu/dissertations/970
[9] K–12 Computer Science Framework Steering Committee. 2016. K–12 Computer Science Framework. http://www.k12cs.org/
[10] Giselle O Martin-Kniep. 2000. *Becoming a better teacher: Eight innovations that work.* ASCD.
[11] Anne Meyer, David H Rose, and David Gordon. 2014. Universal design for learning: Theory and practice. *(No Title)* (2014).
[12] Django Paris. 2012. Culturally sustaining pedagogy: A needed change in stance, terminology, and practice. *Educational researcher* 41, 3 (2012), 93–97.
[13] Carol Ann Tomlinson. 2000. Differentiation of Instruction in the Elementary Grades. ERIC Digest. (2000).
[14] Grant P Wiggins and Jay McTighe. 2005. *Understanding by design.* ASCD, Topsham, ME, USA.
[15] Dazhi Yang, Youngkyun Baek, Yu-Hui Ching, Steve Swanson, Bhaskar Chittoori, and Sasha Wang. 2021. Infusing Computational Thinking in an Integrated STEM Curriculum: User Reactions and Lessons Learned. *European Journal of STEM Education* 6, 1 (2021), 4. https://doi.org/10.20897/ejsteme/9560

Scaling Collaborative Learning: Using the Community Digital Library to Enrich Course Content

Kevin Ros
kjros2@illinois.edu
Computer Science
University of Illinois Urbana-Champaign
Urbana, Illinois, U.S.A.

ChengXiang Zhai
czhai@illinois.edu
Computer Science
University of Illinois Urbana-Champaign
Urbana, Illinois, U.S.A.

ABSTRACT

The ability for a student to discover and learn from online material outside of a course's core curriculum is critical for generalized understanding. However, such a process is often limited by various factors, including efficiency, redundancy, feasibility, and a lack of domain knowledge. We propose to optimize this discovery process and facilitate student collaboration in seeking online information relevant to the course by leveraging the Community Digital Library (CDL) as a tool for enabling students and instructors to enrich course content by collaboratively indexing, searching, and discovering course-related webpages. As a collaborative learning tool, the CDL also enables instructors to identify topics in the lectures that should be improved over time. We report promising preliminary results using CDL for a graduate course.

CCS CONCEPTS

• **Information systems → Digital libraries and archives**; **Social tagging systems**; *Social recommendation*; *Web indexing*; *Collaborative search.*

KEYWORDS

community digital library; social tagging; collaborative search; web search engines; information retrieval; online learning

ACM Reference Format:
Kevin Ros and ChengXiang Zhai. 2024. Scaling Collaborative Learning: Using the Community Digital Library to Enrich Course Content. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635583

1 INTRODUCTION

Traditional learning structures are commonplace among today's educational environments, and these structures often include instructor-specified core content over which the students progress during the semester (e.g., lecture videos, textbook chapters, etc.). A course's core content selection is seldom complete, as differing student backgrounds and goals can lead to information needs beyond what can be satisfied by the course. To resolve these needs, students may turn to the internet to to gain deeper understandings of the course content. However, the existing information discovery process is problematic, especially with regard to course management and collaboration. The process is often independent among students despite their similar needs.[1] Moreover, a student who lacks background knowledge may find it difficult to formulate efficient queries due to a vocabulary gap [6, 9]. Additionally, the instructor may not receive direct feedback on the topics which students find difficult, thus making it harder to improve the course content over time.

As a first step towards mitigating the issues surrounding online information discovery in an educational setting, we explore the use of the Community Digital Library (CDL) [7] for collaborative learning.[2] The CDL is an online[3] and open-source[4] social bookmarking platform that enables any group of users that share similar information needs to describe and save, search for, and discover online content in a collaborative way. A complete overview of the CDL can be found on the aforementioned website and the paper [7].

We performed a case study by using the CDL in a large, graduate-level hybrid classroom during the spring semester of 2023 and found it to be useful. Our proposed methodology is to create an assignment where students are asked to use the CDL to save, per lecture, a webpage that they found helpful for understanding the lecture content. Submitted websites are saved to a course-wide community, with each submission referencing the corresponding lecture (via a hashtag), and containing a title and description describing the web resource's utility to the lecture. As a result, students and instructors are able to leverage the CDL to work towards mitigating many of the aforementioned problems regarding information discovery in classrooms. Our proposed methodology is inspired by prior studies of social bookmarking in education [2, 4, 5, 8]. Distinct from the cited studies, we study a new tool and focus our analysis on the content of the submissions themselves (i.e., the commonly-submitted domains, topics, keywords, and interaction logs).

2 THE CDL AND COLLABORATIVE LEARNING

The CDL, being a general platform, has the ability to support collaborative information seeking in numerous ways. In the simplest setting, members of a group who wish to collaborate can form a community on the CDL, and then they can make submissions to this community. Each submission consists of a webpage URL, a title, and a description. In the title and description, a user can

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635583

[1] As the old adage goes: If you have a question, then somebody else probably has it too!
[2] In the time between when this study was conducted and published, the CDL was renamed to TextData.
[3] https://textdata.org/
[4] https://github.com/thecommunitydigitallibrary/cdl-platform

explain how the webpage is useful for the members of the community. As the community content grows, each member can browse or search for content according to their community-related information needs. Moreover, the phrases used in the user-provided titles and descriptions can help community members find content in ways that go beyond the content from the original web resource. Beyond traditional search and browsing, the CDL also supports hashtag-based search. If a submission title or description contains a hashtag, then any search in that community with that hashtag will return the submission. In this way, community members can precisely filter their searches (e.g., according to a specific lecture or goal). Even if the community contains thousands of web resources, community members can still effectively find relevant information given sufficient hashtags. While there are several other popular education-based social bookmarking platforms (such as Diigo [3] or Pinboard [1]), the CDL is one of the only online, open-source, and actively developed social bookmarking platforms, thus it allows for easier adoption by instructors. We report and analyze our positive experience using the CDL in an educational context.

3 FINDINGS

Throughout the semester, a total of 984 submissions were made by the 41 consenting students, which roughly corresponded to the required 21 submissions per student. Our main findings were that, through this simple assignment, students submitted a wide breadth of diverse content, both in topic and in medium, covering the main course topics, with trends towards the more difficult topics. Moreover, the submissions were immediately useful for other students, as evidenced by submission descriptions and by the query search and browsing metrics. The submissions also offered valuable insights for the instructor, who can immediately use the submitted webpages to improve the lectures for future semesters. Regarding the submission sources, we found that the submission domains and URLs were very diverse, with 287 unique domains out of the 984 submissions. Moreover, only 58 URLs were submitted twice. The submissions also covered a wide range of content mediums (e.g., blog posts, tutorials, videos, reference material, code repositories, papers. etc.). There was a significant amount of diversity among the titles and descriptions across the submissions. The students described the utility of the submissions using both terminology from the course topics (e.g., "language model", "em algorithm") and helpful meta-descriptors (e.g., "example", "application", "introduction"). Moreover, the submissions both covered course topics and went beyond to concepts not mentioned in the lectures.

Through some basic log data analysis, we found that students used the CDL for searching, revisiting, and browsing the course community submissions. The primary mode of information discovery was searching using the search bar, and included both hashtag-based filtering and keyword-based queries. There were 907 queries submitted via the search bar and 298 submission clicks. Moreover, users viewed their own submissions 412 times and the submissions of the course community 374 times. Across both cases, a result was clicked 69 and 94 times, respectively.

There were also some challenges and limitations regarding our case study. Some challenges included ensuring that the student-submitted content was of high quality and correct, encouraging the use of the tool beyond the course-base assignment, and properly assessing the utility of the tool beyond the log-based interactions. And regarding the limitations, with the study taking place over a single course during one semester, it was difficult to draw general conclusions that will extend to other courses, especially if the format or content is different. We can address this limitation by performing additional studies. In particular, the content saved for a course via the CDL in one semester can be naturally reused and extended by the future students of the same course, enabling convenient additional studies to address this limitation.

4 CONCLUSION AND FUTURE WORK

Through our preliminary case study, we found that the CDL is a promising tool for supporting collaborative learning. With a simple assignment, students were able to collect a wide breadth of content relating to the lectures. In the future, we plan on performing further analysis on both the quality of the submissions and the quality of their corresponding webpages. We will investigate ways for students and instructors to effectively engage with the dataset and utilize the web resources in the course, beyond just traditional keyword-based and hashtag-based search. For example, we will study ways of automatically organizing submissions around common community topics and ways to incorporate user interactions in real time to rank submission quality.

Our proposed use of the CDL for helping scale online learning by enriching course content can be applied to any course regardless of its format. The instructions for the assignment used by us are available on the CDL website.[5] The instructions can be reused or adapted by any instructor for deployment of the same or a similar assignment for their courses. Moreover, because the CDL is open source, it provides a great opportunity for researchers, developers, and industry practitioners to further customize it towards a desired use case. We look forward to seeing how this case study helps guide others towards leveraging the CDL for their particular needs.

REFERENCES

[1] Maciej Cegłowski. 2023. *Pinboard.* https://pinboard.in/
[2] Clara Coutinho and João Bottentuit Junior. 2008. Using social bookmarking to enhance cooperation/collaboration in a teacher education program. In *EdMedia+ Innovate Learning*. Association for the Advancement of Computing in Education (AACE), 2551–2556.
[3] Diigo.com. 2023. *Diigo: Your Learning, Simplified.* https://www.diigo.com/
[4] Enrique Estellés, Esther Del Moral, and Fernando González. 2010. Social bookmarking tools as facilitators of learning and research collaborative processes: The Diigo case. *Interdisciplinary Journal of E-Learning and Learning Objects* 6, 1 (2010), 175–191.
[5] Tricia M Farwell and Richard D Waters. 2010. Exploring the use of social bookmarking technology in education: an analysis of students' experiences using a course-specific Delicious. com account. *MERLOT Journal of Online Learning and Teaching*. 6, 2 (2010).
[6] Hang Li, Jun Xu, et al. 2014. Semantic matching in search. *Foundations and Trends® in Information Retrieval* 7, 5 (2014), 343–469.
[7] Kevin Ros and ChengXiang Zhai. 2023. The CDL: An Online Platform for Creating Community-based Digital Libraries. In *Companion Publication of the 2023 Conference on Computer Supported Cooperative Work and Social Computing*. 372–375.
[8] Nashrawan Taha, Jamie Wood, and Andrew Cox. 2016. Social bookmarking pedagogies in higher education: A comparative study. In *Professional Development and Workplace Learning: Concepts, Methodologies, Tools, and Applications*. IGI Global, 1420–1433.
[9] Christophe Van Gysel. 2017. Remedies against the vocabulary gap in information retrieval. *arXiv preprint arXiv:1711.06004* (2017).

[5] https://textdata.org/documentation under "Use Cases > Lecture Content Expansion".

Using Generative AI to Support PK-12 Teaching and Learning:

Developing Sample Lessons and More

Pati Ruiz
Learning Sciences Research
Digital Promise
Redwood City, CA USA
pruiz@ digitalpromise.org

Alessandra Rangel
Learning Experience Design
Digital Promise
Redwood City, CA USA
arangel@digitalpromise.org

Merijke Coenraad
Learning Experience Design
Digital Promise
Washington, DC USA
mcoenraad@ digitalpromise.org

ABSTRACT

North Salem Central School District (North Salem) has worked with researchers as part of a larger Research Practice Partnership (RPP) to design and implement an inclusive PK-12 computing pathway in their district. This poster describes how teachers used Generative AI (GenAI) tools in three areas: (1) the development of sample computational thinking (CT) lesson plans; (2) initial brainstorming; and (3) professional learning.

As North Salem reflected on their use of GenAI tools, they named two AI tools specifically: OpenAI's ChatGPT-4 and Bing's Image Creator. Teachers also describe ethical dilemmas that they faced when integrating GenAI tools as well as other concerns that will be described below. This work builds on the growing literature on the use of Generative AI tools to support the day-to-day work of teachers.

CCS CONCEPTS

• Applied computing → Social and professional topics → Professional topics → Computing education → K-12 education, Computing literacy

KEYWORDS

K-12 computer science education, educational equity, formative assessment

ACM Reference format:

Pati Ruiz, Alessandra Rangel and Merijke Coenraad. 2024. Using Generative AI to Support PK-12 Teaching and Learning: Developing Sample Lessons and More. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635522

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635522

1 INTRODUCTION / PROBLEM

"Generative AI can be a tremendous opportunity for human development, but it can also cause harm and prejudice. It cannot be integrated into education without public engagement, and the necessary safeguards and regulations from governments. This UNESCO Guidance will help policymakers and teachers best navigate the potential of AI for the primary interest of learners." - Audrey Azoulay, UNESCO Director-General

After the initial banning access to Generative AI in the United States and around the world[1], guidance from organizations like UNESCO has begun to provide some guidance to governments, and schools as they explore the ethical use of Generative AI while protecting student, teacher, and community data and privacy standards[2]. and all school districts are following suit. Some educators believe that these AI systems and tools are out in the world and the best thing educators can do is to teach students to partner with AI tools, so they can be better prepared for a technological world. For example, English teacher Cherie Shields was recently interviewed by the New York Times where she shared that she assigned students in one of her classes to use Chat GPT to create outlines for a recent essay assignment. She shared that the process helped deepen students' understanding of the stories while also teaching them to interact with an AI system by manipulating their inputs to get the responses they were looking for. In this case, ChatGPT became a tool that can support learning when we thoughtfully include it in our lessons and also guide students in using it well.

Dr. Julio Vazquez, Director of Instruction and Human Resources, and his team are encouraging experimentation and access to ChatGPT for all faculty and staff and are thinking about how to provide students with access in a manner that will not conflict with student privacy laws. Staff members are rolling their sleeves up and starting to explore and learn about how they can use it with their students productively. In fact, they are exploring the use of ChatGPT to develop sample Computational Thinking (CT) lesson plans that the team uses as a jumping off point in their CT Pathways development process.

Dr. Vazquez and his colleague Cynthia Sandler, Library Media Specialist, started by entering prompts into ChatGPT and seeing what came out. The searches started with prompt terms that went something like "*generate a 5th grade lesson for computational thinking focusing on science.*"

As the team began to analyze the lesson plans that came out, they realized they needed to make adjus2tments. Julio shared that he and his team have become better at giving ChatGPT enough context so that the lessons that are developed are closer to what the team expects of a lesson plan and the content better aligns to both CT and content area standards. For example, a more recent lesson prompt terms included: "write a science lesson that integrates 9-12.CT.1." and "Create a simple digital model that makes predictions of outcomes. and HS-PS1-5." And "Apply scientific principles and evidence to explain how the rate of a physical or chemical change is affected when conditions are varied."

2 OVERVIEW / METHODS / RESULTS

This poster reports on the analysis of 60 teacher surveys, focus group data collected from 5 teacher participants, 1 interview with a district administrator, and the sample lesson plan artifacts themselves examined by two researchers and one practitioner in the RPP. The 5 teachers who participated in the focus groups taught Media Arts, English as a New Language, Speech and Language Pathology in grade levels K-5 and 6-12. The data was completed between April and July 2023. North Salem was well positioned to integrate Generative AI in the form of Chat GPT because the team has experience co-designing CT lesson plans and artifacts with researchers and they have been working with the research team for four years. Dr. Vazquez described Generative AI like this: "ChatGPT is a good prompt for conversation. I see this tool as a starting point for teachers and students." For the Inclusive CT Pathways team this meant recognizing early on that the group needed more time to understand what the terms equity and inclusion meant in their context. After some deep thinking and work on operationalizing terms like equity and inclusion within the context of Talladega, the group developed equity-centered rubrics and "look-for" documents for three overarching CT competencies found here [blinded link]. The intentional integration of inclusivity within the district was applied to the formative assessment artifacts by validating nonconventional ways of doing and knowing computing including the flexibility to express their interests and experiences in computational artifacts.

3 CONTRIBUTIONS AND FUTURE WORK

This work contributes to a growing literature about operationalizing inclusivity practices for computing education, particularly formative assessment. We hope these examples of inclusive computing artifacts will inspire teachers, coaches, and administrators to reflect on the equity goals and needs of their own districts and intentionally integrate inclusivity into their own similar artifacts as they move beyond access to asset-based computing education opportunities

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under grant numbers 2219350 and 2112635.

REFERENCES

[1] Michael Elsen-Rooney. 2023. NYC Education Department blocks CHATGPT on school devices, Networks. Chalkbeat New York. Retrieved October 11, 2023 from https://ny.chalkbeat.org/2023/1/3/23537987/nyc-schools-ban-chatgpt-writing-artificial-intelligence.

[2] Fengchun Miao and Wayne Holmes. 2023. Guidance for generative AI in education and research. UNESCO, Paris. ISBN 978-92-3-100612-8. Retrieved October 11, 2023 from https://unesdoc.unesco.org/ark:/48223/pf0000386693.locale=en

[3] Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI:https://doi.org/10.1007/3-540-09237-4.

[4] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY.

Community Building and Joyful Learning in Computer Science Education Through Online At-Scale Seminars

Ana Mary Rusch
ARusch3@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Eric Ianni*
ianni@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

ABSTRACT

Seminars change the landscape of Computer Science education by allowing for a non-traditional space of learning that employs synchronicity, community, and joyful engagement. This paper will discuss the benefits of seminars that are one-credit, pass/fail, and do not count toward graduation, in the Online Master of Science in Computer Science (OMSCS) program at Georgia Tech. Since these seminars are on a pass/fail grading scale and do not count towards graduation, they produce a low-stress environment of learning that allows students to focus on the joy of learning rather than commodified learning focused on assignment production and grades. These seminars are also more flexible than traditional three-credit courses and employ a wide range of pedagogical frameworks. Furthermore, they also address the most significant drawback to online at-scale education: a lack of sense of community. Many of the seminars are participation based and thus encourages community building with scheduled synchronously biweekly meeting and/or asynchronous engagement.

CCS CONCEPTS

• **Applied computing** → **Education**; **Collaborative learning**; **Interactive learning environments**; **Distance learning**; **E-learning**.

KEYWORDS

Computer Science Education, Seminars, Online Learning, Community Building, Joyful Learning

ACM Reference Format:
Ana Mary Rusch and Eric Ianni. 2024. Community Building and Joyful Learning in Computer Science Education Through Online At-Scale Seminars. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635478

1 INTRODUCTION

In the Online Master of Science in Computer Science (OMSCS) program at Georgia Tech, in addition to the standard three-credit courses, one-credit seminars that are on a pass/fail grading system and do not count towards graduation are also offered. One of the authors of this paper helps manage the program's seminars, and both of the authors lead several seminars as individual instructors and as co-instructors. Since their debut in the Fall 2021 until the most recent Summer 2023 semester, there has been a total of 28 seminars offered with 15 unique topics. These seminars were partly inspired by the "Synchronicity Paradox" [4] that points to the interesting findings that students in an asynchronous program joined the program because of its asynchronous model, yet once in the program wanted some kind of synchronicity. Thus, the seminars were created in part to fulfill the desire for synchronicity. However, in our experience, we found that seminars fill more than the synchronous gap in online at-scale learning; they built social communities and reclaim joyful learning in higher education.

Students run the risk of burnout [5], especially in graduate school, which is known to produce stress and contribute to the decline of student mental and emotional health [7]. Moreover, for students in higher education, learning has become synonymous with extrinsic rewards such as grades [9]. In fact, in our experience as instructors, the most common questions students ask us at the beginning of a semester are all related to grades: what is the quantifiable worth of each assignment, and what is the grading scale for the course. Learning for the sake of learning has been lost in the "growing focus on metrics to measure the performance of students, academics, and institutions" [9, p. 272]. Thus, our seminars attempt to reclaim the art of learning as they do not count towards graduation and are on a simple pass/fail grading system; this allows for a low-stress, intrinsically rewarding, and joyful educational experience [9]. Furthermore, the subjects of the seminars promote interdisciplinary thought and pedagogy in computer science education by examining fields such as Sociology, Anthropology, and Philosophy. These fields are explored and rendered central to the theory and practice in computing rather than just in the periphery.

In this same vein, seminars such as *Computational Sociology* not only highlight issues such as ethics and biases in technology and computing but also contributes to diversity, equity, and inclusion through inclusive subject matter, culturally responsive curriculum, and the creation of communities of belonging.Seminars such as the *Women in Tech* seminar create a much-needed safe space of community and discussion on topics that are not typically held and rendered invisible in computer science education. Furthermore, course-based seminars such as the *Computing in Python* provide foundational education and skill building in a low-risk setting for students that might not have extensive experience with this ubiquitous programming language.

*Both authors contributed equally to this research.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635478

2 JOYFUL LEARNING

The pedagogical theory and practice of joyful learning have traditionally been explored in research related to K-12 institutions[1–3, 8], however, research by Heljakka [3] and Nørgård, Toft-Nielsen, and Whitton [9] on playful learning in higher education is a testament to the notion of *play* being beneficial to students of all ages. In this paper, we use the term joyful learning instead of playful learning because we want to focus on the emotion of joy and not conflate it with gamification.

In the *Women in Tech* seminar, for example, a plethora of emotions are felt and expressed throughout the semester. All students, regardless of gender, are welcome to join this seminar, and students from different backgrounds from across the world join every semester. Though they are a group of diverse individuals with different narratives and life stories, a common thread that knits our community is the support we gift each other as we learn and discuss the history as well as the current topics affecting women and nonbinary technologists. In order to facilitate increased representation in computer science, the instructor of this seminar also invites diverse guest speakers from academia, NGOs, and industry to network with the students and share their own lived experiences. As mentioned above, this seminar produces a variety of emotions, even tears, yet we consider this a form of joy as the ability to express these emotions in a safe space is both healing and joyful. Furthermore, as a woman in computer science education, the instructor of this seminar relates to the discussions students have as they mirror her own lived experiences. Thus, this seminar provides an affective educational experience that the instructor shares alongside her students. This shared experience between instructor and students is another factor that promotes joyful learning [9, p.276].

Furthermore, in both the *Futurism* and *Computing and Popular Culture* seminars, the seminars were designed with joy at the center. The students are asked what their interests are, even if they lay outside of the field of computer science, and then the instructors incorporate those interests to make the content both relevant and joyful to learn. Moreover, the instructors assign various academic articles, short stories, and even more mainstream media, such as feature films and music in these seminars. This way, the seminar explores the field of computer science through different modalities of academia and art. As an example, students can begin by reading about the development and ethical issues of artificial intelligence, robotics, and surveillance systems but then examine the same topics through popular cultural depictions of the same that inspires different ranges of emotion. Strategies such as these can often be seen as not rigorous [9] and might not be readily employed in traditional courses, but seminars have the freedom to explore different pedagogical strategies.

3 BRIDGING THE GAPS

Lack of community, social disconnectedness, and loneliness are inherent risks in online learning [4][10]. However, scholars such as McInnerney and Roberts [6] and Yuan and Kim [10] have pointed out that synchronicity is key in establishing online social communities. These social communities are primordial in combating the associated risks with online education. However, in programs such as ours that is both online and at scale, synchronicity in traditional three-credit courses is not always possible and not always accessible for students. Thus, seminars satisfy a synchronous component while also being flexible in the sense that they are optional and not a program requirement. Furthermore, seminars also have asynchronous components through Ed Discussion forums for the students that still want to engage in the seminar but rather do it asynchronously.

Developing a sense of community and belonging in an online at-scale program can be difficult, and that is where our seminars come in. They afford students a synchronous option to gather together for a mutual goal in a low-risk environment. While the subject of each seminar may different, they all allow students a safe space to explore the subject and connect with each other. Collaborative seminars, such as the ones in our program, create an environment where students are able to express their thoughts and to further develop skills in a supportive space where their graduation requirements are not at stake. Moreover, beyond adding a synchronous element to our program, we also wanted a platform to further build community and sense of belonging. Anecdotally, based on comments made in the synchronous meetings and in the chats, the instructors of the seminars saw that attendees felt a stronger sense of community by attending the seminars. In fact, the students in *Computing and Popular Culture* seminar did not want the seminars to end and made the request to either kept meeting biweekly or create a a club based on the seminar. Similarly, a Slack channel for the *Women in Tech* seminar was created by a student from the seminar to keep that connection with the community and discussion going even after the seminar had ended.

REFERENCES

[1] Subuh Anggoro, Wahyu Sopandi, and M Sholehuddin. 2017. Influence of joyful learning on elementary school students' attitudes toward science. In *Journal of Physics: Conference Series*, Vol. 812. IOP Publishing, 012001.

[2] Hilary G Conklin. 2014. Toward more joyful learning: integrating play into frameworks of middle grades teaching. *American Educational Research Journal* 51, 6 (2014), 1227–1255.

[3] Katriina Heljakka. 2023. Building playful resilience in higher education: Learning by doing and doing by playing. In *Frontiers in Education*, Vol. 8. Frontiers, 1071552.

[4] David A Joyner, Qiaosi Wang, Suyash Thakare, Shan Jing, Ashok Goel, and Blair MacIntyre. 2020. The synchronicity paradox in online education. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*. 15–24.

[5] Boram Kim, Sooin Jee, Joungwha Lee, Sunghee An, and Sang Min Lee. 2018. Relationships between social support and student burnout: A meta-analytic approach. *Stress and Health* 34, 1 (2018), 127–134.

[6] Joanne M McInnerney and Tim S Roberts. 2004. Online learning: Social interaction and the creation of a sense of community. *Journal of Educational Technology & Society* 7, 3 (2004), 73–81.

[7] Maral PS Mousavi, Zahra Sohrabpour, Evan L Anderson, Amanda Stemig-Vindedahl, David Golden, Gary Christenson, Katherine Lust, and Philippe Buhlmann. 2018. Stress and mental health in graduate school: How student empowerment creates lasting change. *Journal of Chemical Education* 95, 11 (2018), 1939–1946.

[8] Linda F Nathan. 2020. Joyful learning at scale: Immersing students in the arts. *Phi Delta Kappan* 101, 8 (2020), 8–14.

[9] Rikke Toft Nørgård, Claus Toft-Nielsen, and Nicola Whitton. 2017. Playful learning in higher education: developing a signature pedagogy. *International Journal of Play* 6, 3 (2017), 272–282.

[10] J Yuan and C Kim. 2014. Guidelines for facilitating the development of learning communities in online courses. *Journal of Computer Assisted Learning* 30, 3 (2014), 220–232.

Middle School CS Curriculum and Standards Alignment

Daisy Rutstein
dazerw@yahoo.com
edCount, LLC

Satabdi Basu
satabdi.basu@sri.com
SRI International

Hui Yang
hui.yang@sri.com
SRI International

Arif Rachmatullah
arif.rachmatullah@sri.com
SRI International

Carol Tate
carol.tate@sri.com
SRI International

ABSTRACT

The development of the CS content standards underscores the importance of curricula aligned with the standards, ensuring equitable coverage of CS concepts for all students. Because standards are broad, we emphasize the need for CS curricula to specify not only the standards they align with but also which aspects of the standards they align with and how. We map one common middle school CS curriculum to a few standards to demonstrate this need.

ACM Reference Format:
Daisy Rutstein, Satabdi Basu, Hui Yang, Arif Rachmatullah, and Carol Tate. 2024. Middle School CS Curriculum and Standards Alignment. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635546

1 INTRODUCTION

As the demand for CS professionals and computationally literate citizens grows, the number of CS offerings in K-12 schools has significantly expanded. Quality CS curriculum is a cornerstone of equitable CS education and determines what CS constructs students will learn and how they will learn the target constructs. However, there are significant disparities in access to quality CS curricula in the U.S. due to various factors including the fact that curriculum decisions are being made at district, school, or even classroom levels. CS standards alignment can provide an objective way to compare different CS curricula and determine their quality. However, mapping CS standards [1] to curricula is complex as the standards are often too broad and open to interpretation [2]. Standards outline the target content and practices and, in theory, support a coherent approach across grade levels, but their high-level descriptions make it challenging to use them as a guide for curriculum selection.

When CS curricula specify standards alignment, it is not very meaningful without a description of how lessons align with the standard and which parts of the standard lessons align with. Towards this end, our paper seeks to explore the standards alignment specified by a well known middle school CS curriculum and how well it actually enables student learning of concepts emphasized in the middle school CS standards for algorithms and programming.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635546

2 BACKGROUND

The landscape of middle school CS education currently includes several CS curricula, some common ones being Code.org's CS Discoveries, Google's CS First, CodeHS's middle school CS pathways, Project STEM's CS Explorations, and CS Unplugged by Colorado School of Mines. Most of these curricula describe alignment with either the K-12 CS Framework or the Computer Science Teachers Association (CSTA) CS standards [1]. However, the descriptions do not include which aspects of the standards their curricula address or how their curricula address the standards. While standards alignment may not be the only criterion education leaders and teachers use for curriculum selection, this lack of detailed information makes the selection process particularly challenging.

In this paper, we focus on code.org's Computer Science Discoveries (CSD) curriculum [5], designed to captivate and engage middle school students with no to limited prior CS background. The Code.org 2021 Annual Report [4] mentions that 788,377 students began using CSD that academic year (approximately 41% were female and 48% belonged to racial and ethnic groups underserved in STEM/CS). Based on these statistics, CSD is considered as one of the most widely embraced middle school CS curricula in the US, and thus seems like an appropriate choice for starting to explore standards alignment of CS curricula.

3 METHODS

In this paper, we focus on five middle school CSTA standards from the 'Algorithms and Programming' (AP) strand (2-AP-10 through 2-AP-14) covering the concepts of algorithms, variables, combined control structures, decomposition, and procedures, respectively. We used an Evidence-Centered Design approach [7]to break down each standard into meaningful and granular learning targets (LTs). These LTs were based on current literature related to the standards and prior documents that unpacked the concepts in the standard [3, 6]. The LTs went through a round of expert review by 4 external K-12 CS education experts and refinement based on the expert feedback.

Once the LTs were finalized, we examined the alignment between our five target standards and the CSD curriculum (Units 1-6A) by focusing on the alignment of LTs to individual CSD lessons. For each standard, we studied its alignment with the CSD curriculum along three dimensions: (1) the number of lessons that aligned with the standard in some way, (2) the mapping between specific LTs for the standard and specific CSD lessons, and (3) the strength of alignment between aligned LTs and lessons.

For each lesson, we started by noting the standards that the curriculum developers identified as aligned with the lesson. Next, we went through the set of activities students engage with during

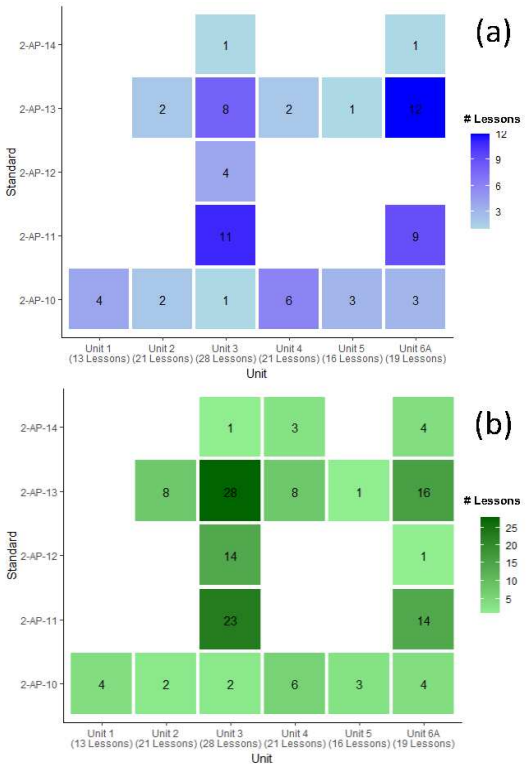


Figure 1: Number of CSD lessons aligned with the target standards: a) Only directly aligned lessons; b) Directly and indirectly aligned lessons

the lesson to determine if any of the activities relate to any of the LTs across all five standards. We identified two levels of alignment, direct and indirect. A direct alignment indicates that students received instruction that was directly related to the LT. For example, when students had an activity where they identified problems with variable names, this was directly aligned with our 2-AP-11 LT related to the naming of variables. For an indirect alignment, students may have opportunities to engage with the LT but it may depend on how the teacher frames the activities or how the students design their projects. For example, alignment with particular LTs that focus on control structures for lessons in which students are given the freedom to pick their own projects may depend on which control structures, if any, students decide to use. We used a lesson's alignment to LTs to determine its overall alignment to a standard. If a given lesson was directly aligned to any LT for a standard, we stated that the overall standard was directly aligned to the lesson. If the only LT alignments for a standard were indirect alignments, we called the overall standard indirectly aligned to the lesson. If no LTs for a standard were specified as either direct or indirect alignment, the standard was deemed not aligned to the lesson.

4 FINDINGS AND DISCUSSION

Figure 1 illustrates the alignment of CSD units 1-6 with our five target standards. CSD units that did not focus on programming concepts (units 1, 2, 4, and 5) were only aligned with 2-AP-10 (algorithms) and/or 2-AP-13 (decomposition) and were not aligned with the programming standards. Overall, we found that all CSD units covered at least one of our five target standards, though alignment was limited to specific aspects of standards. For example, Unit 1 focused only on recognizing relevant information to create an algorithm, but did not cover other aspects such as creating flowcharts or comparing or testing algorithms. Unit 3 covered all standards but focused only on specific aspects such as knowledge that algorithms need to be precise, simple variables (did not include compound variables such as lists), compound conditionals (did not include nested loops or nested conditionals which are called out in the standard), and creating and calling procedures without input parameters.

Discussion. Currently, most CS curricula indicate their standards alignment. We point out that the usefulness of this approach is limited if curricula do not specify how they align with standards and which aspects of standards lessons align with. Using the CSD curriculum, we demonstrate what detailed standards alignment can reveal about the content coverage of a curriculum. Examining alignment with only five CS standards, we find that while CSD covers all the standards to varying degrees, it does not provide complete coverage of all components of the standards. This is not an indication of the quality of the curriculum, but more a recognition that the CS standards are complex and cover a lot in one standard [2]. Additionally, CS curricula typically need to cover some precursors to the grade-level standards, increasing the number of concepts to cover. A detailed and more transparent account of standards alignment can help school leaders make informed choices when selecting curricula and can indicate to teachers that they need to supplement student learning on certain concepts. Our paper demonstrates the value of expanding this work to unpack more CS standards and examine standards alignment for additional curricula.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. DRL-2010591. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Computer Science Teachers Association. 2017. CSTA K-12 computer science standards. Revised 2017. http://www.csteachers.org/standards
[2] Satabdi Basu, Daisy Rutstein, Carol Tate, Arif Rachmatullah, and Hui Yang. 2022. Standards-Aligned Instructional Supports to Promote Computer Science Teachers' Pedagogical Content Knowledge. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*. 404–410.
[3] Satabdi Basu, Daisy W Rutstein, Yuning Xu, Haiwen Wang, and Linda Shear. 2021. A principled approach to designing computational thinking concepts and practices assessments for upper elementary grades. *Computer Science Education* 31, 2 (2021), 169–198.
[4] Code.org. 2021. 2021 Annual Report. https://code.org/files/code.org-annual-report-2021.pdf
[5] Code.org. n.d. Computer Science Discoveries. https://code.org/educate/csd
[6] Shuchi Grover and Satabdi Basu. 2017. Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education*. 267–272.
[7] Robert J Mislevy and Geneva D Haertel. 2006. Implications of evidence-centered design for educational testing. *Educational measurement: issues and practice* 25, 4 (2006), 6–20.

Elementary Latinx Students Apply Growth Mindset while Creating in Scratch

Dana Saito-Stehberger
dsaitost@uci.edu
University of California, Irvine
Irvine, CA, USA

Jennifer K. Houchins
jhouchi@wested.org
WestEd
San Francisco, CA, USA

Mark Warschauer
markw@uci.edu
University of California, Irvine
Irvine, CA, USA

ABSTRACT

The Latinx population is underrepresented in secondary computer science (CS) courses as well as in the STEM workforce. By exposing Latinx students to positive coding experiences and encouraging a growth mindset earlier in school, Latinx participation in CS may likely increase. This poster presents the results of examining the reflections upper-elementary students recorded in their journals after completing a structured Scratch coding project. Students were prompted to share problems they faced, how they responded to those problems, and what they were proud of in their project.

ACM Reference Format:
Dana Saito-Stehberger, Jennifer K. Houchins, and Mark Warschauer. 2024. Elementary Latinx Students Apply Growth Mindset while Creating in Scratch. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635568

1 INTRODUCTION

Tremendous progress to broaden the instruction of computing in K-12 contexts is being made. In a span of four years (2017-2021), the number of states that have secured funding for computer science went from 9 to 31 in the United States [3]. Computing and computer science (CS) have become compulsory topics in primary schools in countries, such as Finland, England, and Estonia [5]. Students are being empowered to be creators of technology, not simply users. Because CS courses in secondary school are electives and students must choose to enroll, it is vital for students to have positive computing experiences earlier in their education. In 2021, only 16% of the high school AP Computer Science test takers were Latinx, while 27% of the general US high school population were Latinx [3]. Following this trend, Latinx workers are also underrepresented in the STEM workforce. They make up 17% or the total employment in the United States, but just 8% of all STEM workers [6]. To create a balanced representation of Latinx individuals in the CS field, it is essential to provide more compelling computing learning experiences to underrepresented students at the elementary grade levels. In addition to attracting underrepresented groups to CS classrooms and careers, primary schools need to equip these students with persistence and a sense of belonging to enhance the tendency to remain. One way schools are addressing this need is by teaching students to have a growth mindset, the belief that one's current capabilities can be developed [4]. Studies support that a growth mindset can be a useful tool to support the learning of students from different groups [2].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635568

2 OVERVIEW OF THE CURRICULUM

The ECforALL curriculum is an introductory, project-based Scratch curriculum that is designed to teach computational thinking to multilingual learners and culturally and linguistically diverse students in upper-elementary school. Projects and learning activities aim to develop students' computing abilities as well as their growth mindset, the belief that they can develop their abilities with dedication and hard work. The Name Poem and the About Me project guidelines encourage students to brainstorm some of their qualities and interests about themselves that they want to share with others. In order to expose students to STEM careers that they may not otherwise be aware of, there are four Role Model lessons that present videos about someone who is from an underrepresented group in the field of computing. Each video provides something about the person that students may be able to relate to, information about their job, and what can be learned from that person. Lastly, after each project is completed, students complete a Reflection Journal where they reflect on a computational concept, an aspect of the creation process, a problem they faced and how they solved it, and something from the project that they are proud of.

3 RESEARCH DESIGN

The aim of this study is to analyze students' reflections about problems they faced after creating their first structured Scratch project in class. The research questions framing this study were:

(1) What challenges did 4th grade Latinx students face when creating their first structured Scratch project?
(2) What are some solutions that worked for the students?
(3) What did students learn that they were proud of?

Eighty-eight 4th grade students in three classrooms completed the Reflection Journal after creating their unique Name Poem project. These classrooms were in a district comprised primarily of Latinx students (98%) and English language learners (60%). Students responded to these two prompts:

(1) "What problem did you face while creating this project? How did you respond to the problem?"
(2) "What is something you did or learned in this unit that you are proud of?"

Sentence frames were provided to help language learners articulate their ideas. One class wrote their answers in a journal while another class digitally submitted their answers via a Google Form.

A third class had students submit their answers both manually in their workbook as well as digitally via Google Form.

To answer the research questions, the responses were analyzed for emerging themes. The authors together followed the thematic analysis steps [1] by first iteratively reading the responses and noting initial ideas. Then, the initial codes were created and relevant responses were assigned to the new themes.

4 FINDINGS & CONTRIBUTIONS

The first research question inquires about the problems students experienced as they coded their first project in the curriculum. We grouped the responses into three general categories. The first category was **"problems that arose from not knowing how to navigate the Scratch environment."** Of the 88 responses, 66 fit into this category. Of those 66 responses, 25 mentioned issues with the movement of the sprite, 10 with editing and coloring the sprite, and 10 with manipulating the costume. Other issues dealt with resetting the sprites, duplicating them, the backdrop, and sound. The second category was **"problems caused by the nature of Scratch."** Three students reported that parts of their project would get deleted or just disappear. Unfortunately, there is not an "undo" button in Scratch that can reliably reverse such an action. The third category was **"problems students had in the development process of their project."** Students were unsure where to put the sprites on the stage or had a hard time deciding what kind of sprites to use. Responses such as *"I didn't know what to make it say"* or *"I had a hard time figuring out what sequence was best for my poem"* fit into the category. Only one student reported having no problems: *"I didn't have a hard time on anithing[sic]."*

The second research question asks students how they solved their problem. There were 37 responses to this question. Twenty-three students reported that they **"asked someone for help"** and 14 reported that they fixed their problem by **"tinkering and moving blocks around."** Both responses reflect students' ability to self-advocate and to seek out viable solutions.

The third research question asked students what they did or learned in the program that they are proud of. Some responses fit into more than one category. Forty-six of the responses reflected pride in knowing how to do something in the Scratch environment, such as change the costume or alter the color of a sprite. Fourteen responses reflected pride in finishing the project. Ten responses fit in the "other" category. Examples of these responses include:

> *"I learned that I love coding."*
>
> *"I learned how to share."*
>
> *"I learned that Scratch is fun."*
>
> *"I learned that I should always listen."*
>
> *"I am proud that I got to help other people with their own poem."*

Lastly, 25 responses reflected a growth mindset perspective. These responses reflect a sense of perseverance and belief that, despite difficulty, they can be successful:

> *"I created a script that was incredible. I can do it. I learned that we can learn new things."*
>
> *"I'm just proud that I was able to do the things I didn't know I was sure of."*
>
> *"I created a script that helps people. I can help. I learned that I can fix anything."*
>
> *"Script is cool. I can do it. I learned that it's hard sometimes."*
>
> *"I learned that you do not get it the first chance."*

5 CONCLUSION

The study presented in this poster examines the problems that Latinx elementary students experience when they are learning to code in Scratch and how they respond to those problems. The most common challenges were related to learning how the Scratch environment works. Other issues were limited functionality of Scratch and challenges of developing ideas for their projects. Students solved their problems with active self-promotion/self-regulated learning by seeking help and tinkering until they found a solution. We also examined what students learned and what they were proud of while creating their projects. Students' reflections showed evidence that students are internalizing growth mindset concepts and believe that they can achieve their goals. Their responses highlight a sense of persistence and that hard work does not seem to have a negative connotation. We are encouraged that students have responded positively to the curriculum and are enjoying the coding process. This growth mindset and positive experience with coding has the potential to be a factor in increasing Latinx participation in secondary CS courses and eventually in the CS field.

ACKNOWLEDGMENTS

We would like to thank the National Science Foundation (Grant 1923136 and 2317832) and the US Department of Education (Grant U411C190092) for providing the funding that made this project possible. Findings expressed in this work are those of the authors and do not necessarily reflect the views of the National Science Foundation or the US Department of Education.

REFERENCES

[1] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.
[2] Susana Claro and Susanna Loeb. 2019. Students with Growth Mindset Learn More in School: Evidence from California's CORE School Districts. Working Paper. *Policy Analysis for California Education, PACE* (2019).
[3] Code.org, CSTA, and ECEP Alliance. 2022. *2022 State of CS Report.* https://advocacy.code.org/stateofcs
[4] Carol S Dweck. 2000. *Self-theories: Their role in motivation, personality, and development.* Psychology press.
[5] Janne Fagerlund, Päivi Häkkinen, Mikko Vesisenaho, and Jouni Viiri. 2021. Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education* 29, 1 (2021), 12–28.
[6] Cary Funk and Mark Hugo Lopez. 2022. Hispanic Americans' trust in and engagement with science. (2022).

Programming Assignment Ungrading as a License to Learn

Implementing Specifications Grading in the Undergraduate Web Development Classroom

Raghav V. Sampangi†
Faculty of Computer Science
Dalhousie University
Halifax, NS, Canada
raghav@cs.dal.ca

Eric Poitras
Faculty of Computer Science
Dalhousie University
Halifax, NS, Canada
eric.poitras@dal.ca

Mayra D. Barrera Machuca
Faculty of Computer Science
Dalhousie University
Halifax, NS, Canada
mbarrera@dal.ca

ABSTRACT

For introductory programming courses, it is crucial to create formative and summarized grading practices that foster growth mindset in students. We explore one way to reimagine and reorient programming assignment grading towards feedback-oriented encouragement for ongoing and continuous learning. This practice of mastery grading encompasses the following key features: (1) students are provided with a comprehensive list of assignment specifications, (2) evaluation of student work is centered on the attainment of specified criteria, employing a nominal scale to denote whether some or all the requirements are met within the deadline, and (3) multiple opportunities are afforded to demonstrate mastery for each specification, without penalties for initial attempts. We share our implementation conducted within an undergraduate first-year web development course, intending it to serve as both a reference and a valuable resource for instructors interested in integrating mastery grading into their own courses.

CCS CONCEPTS

• Social and professional topics ~ Professional topics ~ Computing education

KEYWORDS

Introductory Programming, Mastery Grading, Specifications Grading, Standards-Based Grading

ACM Reference format:
Raghav V. Sampangi, Eric Poitras, and Mayra D. Barrera Machuca. 2024. Programming Assignment Ungrading as a License to Learn: Implementing Specifications Grading in the Undergraduate Web Development Classroom. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635581

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03. https://doi.org/10.1145/3626253.3635581

1 INTRODUCTION

Specifications grading (a.k.a. contract grading), emphasizes a learner's ability to meet specific criteria. This approach is like mastery learning, competency-based grading, and standards-based grading [1], but is focused on learners completing a set of assignments according to pre-defined, descriptive criteria for successful completion. Specifications grading in computer science courses typically employ a binary grading system for each assignment [2], allowing learners to either meet the specified criteria and receive full credit or do not meet them and receive no credit. The overall grade is often determined by the number of tasks or assignments successfully completed. Learners may move on to new content or skills regardless of their performance on previous standards. Feedback is usually provided on a per-standard basis, helping learners and teachers identify specific areas of strength and weakness in relation to learning objectives.

2 RELATED WORKS

Computer science courses employing this grading system have been shown to be rated more favorably by learners, particularly for questions pertaining to assessment approaches [3], while still supporting a large number of students to demonstrate sufficient mastery of the material [6]. Students that went to the follow-on CS2 course showed a higher pass rate than those enrolled to the CS1 course with a traditional grading system [7]. Positive results have also been shown in terms of easing administrative load and time required to mark assignments [4, 5]. Our work is also inspired by the literature on decolonization and post-colonization [8] that explores returning to specification-based, feedback- and apprentice-oriented learning opportunities that seek to integrate practices that are focused on encouraging ongoing and continuous learning, rather than rewards and punishment.

3 IMPLEMENTATION CONTEXT

This work presents a specification-based assignment ungrading practice introduced in a first-year computer science introductory course on web development offered at Dalhousie University. Assignments in this course require students to code web sites based on specifications, beginning with coding simple single-page web pages and websites with minimal styling to coding multi-page accessible and responsive websites that include interactivity implemented using JavaScript. When possible, assignments also require learners to incorporate code generated by AI tools such as

ChatGPT [9]. There are four assignments (A1 – A4) in the course and each assignment – barring A1 – have overlapping learning outcomes with the previous, which allows learners to incorporate feedback into the work for subsequent assignments.

This course has over 300 students enrolled each semester in one or two sections. In previous iterations, these assignments had a significant percentage of the overall course grade, the rationale behind which was to reward student work and ongoing learning. To recognize ongoing learning, we had incorporated ways to reward improvements in subsequent assignments by adding a small percentage of the grade from the subsequent one to the previous if there was a grade improvement. This was still a rewards-focused grading mechanism and we wanted to move to a practice that acknowledged learning in its pure form.

4 STEPS TOWARDS UNGRADING THE ASSIGNMENTS

In the course syllabus, we offer a variety of opportunities for students to showcase their mastery in this course, including timed and untimed tasks, in-class and out-of-class assessments, quizzes, activities, and assignments. Learners in the course were informed that the assessment and grading scheme for assignments could be different in this course as compared to other courses. We also highlighted that we would consider the best 3 performances out of the 4 total assignments, and the 15% assignment grade would be computed as specified in Figure 1. In doing so, students are not penalized for failing to demonstrate mastery on their initial attempt and may reattempt the specification on the next assignment.

Assignment Grading

1. Each assignment is evaluated to assess student work for the learning they have demonstrated, with the objective of providing feedback for iterative and ongoing learning.
2. Assignments have detailed requirements that are organized into functionalities of the web site.
3. For each assignment, a student will:
   a. Get a grade of **Completed & Submitted**, if they have completed all requirements specified and submitted the assignment before the deadline.
   b. Get a grade of **Incomplete & Submitted**, if they have submitted the assignment before the deadline and have only completed some of the requirements, but not all.
   c. Get a grade of **Not submitted**, if they have not submitted the assignment.
4. The final 15% of the assignment grade in the grading scheme will be computed as follows:
   a. Students are eligible to receive the full 15/15 of the assignment grade, if they receive a grade of Complete & Submitted for at least 3 out of 4 assignments.
   b. Students are eligible to receive 10/15 of the final assignment grade, if they receive a grade of Complete & Submitted for at least 2 assignments but not 3.
   c. Students are eligible to receive 6/15 of the final assignment grade, if they receive a grade of Complete & Submitted for at least 1 assignment but not 2, and a grade of Incomplete & Submitted for at least 1 assignment.
   d. Students are eligible to receive 3/15 of the final assignment grade, if they receive a grade of Incomplete & Submitted for at least 3 assignments and have not completed any assignment.
   e. If students do not submit any assignment, they will get a grade of 0/15.

Figure 1: Assignment grading scheme specified in the syllabus

5 REFLECTIONS AND FUTURE WORK

To encourage students to review feedback, we included a post-feedback survey that students were required to complete. This survey included the following two open-ended questions:

1. *Looking at the feedback given to your assignment submission, which aspects will you incorporate into your work in this course for future assignments?*
2. *Are there some aspects of the feedback that you do not plan to incorporate into your practice? What might be some of the reasons leading you to feel this way?*

Our prediction is that student overall stress levels will be lower due to upgrading our practices towards a learning-focused and decolonial model to redirect focus away from academic achievement outcomes to mastery of skills. We will discuss the preliminary findings obtained from these efforts during the poster presentation session, offering insights into both the aspects that have been successful and those that require improvement.

ACKNOWLEDGMENTS

The authors would like to express sincere gratitude to Stephanie Bernier & the CS Ed Team for their feedback and suggestions in the design of the post-survey feedback and for their general support in our teaching and learning endeavours.

REFERENCES

[1] Nilson, L. (2015), Specifications Grading: Restoring Rigor, Motivating Students, and Saving Faculty Time, Stylus Publishing, LLC.

[2] James W. McGuffee, David L. Largent, and Christian Roberson. 2019. Transform Your Computer Science Course with Specifications Grading. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE'19). Association for Computing Machinery, New York, NY, USA, 1234. https://doi.org/10.1145/3287324.3287528

[3] Christian Roberson. 2018. Techniques for using specifications grading in computer science. J. Comput. Sci. Coll. 33, 6 (June 2018), 192–193.

[4] Andrew Berns. 2020. Scored out of 10: Experiences with Binary Grading Across the Curriculum. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20). Association for Computing Machinery, New York, NY, USA, 1152–1157. https://doi.org/10.1145/3328778.3366956

[5] Lijun Chen, Joshua A. Grochow, Ryan Layer, and Michael Levet. 2022. Experience Report: Standards-Based Grading at Scale in Algorithms. In Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1 (ITiCSE '22). Association for Computing Machinery, New York, NY, USA, 221–227. https://doi.org/10.1145/3502718.3524750

[6] Tuson, E. and Hickey, T., 2023, March. Mastery Learning with Specs Grading for Programming Courses. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (pp. 1049-1054).

[7] Albert Lionelle, Sudipto Ghosh, Marcia Moraes, Tran Winick, and Lindsey Nielsen. 2023. A Flexible Formative/Summative Grading System for Large Courses. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569810

[8] Cristaldi, G., Quille, K., Csizmadia, A.P., Riedesel, C., Richards, G.M. and Maiorana, F., 2022, March. The intervention, intersection and impact of social sciences theories upon computing education. In 2022 IEEE Global Engineering Education Conference (EDUCON) (pp. 1561-1570). IEEE.

[9] Becker, B.A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J. and Santos, E.A., 2023, March. Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (pp. 500-506).

Automated Support for Flexible Extensions

Jordan Schwartz
University of California Berkeley
Berkeley, CA, USA
jordanschwartz@berkeley.edu

Madison Bohannan
University of California Berkeley
Berkeley, CA, USA
mkcb@berkeley.edu

Jacob Yim
University of California Berkeley
Berkeley, CA, USA
jacobyim@berkeley.edu

Yuerou Tang
University of California Berkeley
Berkeley, CA, USA
yuerou.tang@berkeley.edu

Dana Benedicto
University of California Berkeley
Berkeley, CA, USA
dbenedicto@berkeley.edu

Charisse Liu
University of California Berkeley
Berkeley, CA, USA
charisseliu@berkeley.edu

Armando Fox
University of California Berkeley
Berkeley, CA, USA
fox@berkeley.edu

Lisa Yan
University of California Berkeley
Berkeley, CA, USA
yanlisa@berkeley.edu

Narges Norouzi
University of California Berkeley
Berkeley, CA, USA
norouzi@berkeley.edu

ABSTRACT

In this work, we present the development of an automated extension tool that supports flexible extension policies. Students interact with a wide range of extension policies in similar ways; in particular, some students repeatedly request multi-day long extensions. When scaled to courses with hundreds or potentially thousands of students, course staff time is the limiting resource preventing adequate student support. We present a tool to help automate a range of extension processes. The use of this tool should reduce staff load while increasing individualized student support, through email communication and consequent recovery of student agency.

Our early research questions are: **RQ1:** Does the extension tool reduce barriers and stigma around asking for assistance? **RQ2:** Does the tool lessen the wait time between requesting and receiving an extension, and how does the tool improve students' learning experience in the course? These questions will help inform us about how an automated tool for flexible extensions helps support growing course sizes and students who may not otherwise receive the support they need for their success and well-being in the course.

KEYWORDS

Accessibility, Equitable and inclusive teaching, Instructional technologies

ACM Reference Format:
Jordan Schwartz, Madison Bohannan, Jacob Yim, Yuerou Tang, Dana Benedicto, Charisse Liu, Armando Fox, Lisa Yan, and Narges Norouzi. 2024. Automated Support for Flexible Extensions. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635628

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635628

1 INTRODUCTION

In response to the growing size of CS courses in colleges and universities [4], the conventional approach of rigid due dates no longer meets the diverse learning needs of students [1, 3]. The rise in mental health issues and global crises has increased the number of students facing exceptional circumstances while pursuing higher education [5]. However, the simultaneous surge in course enrollment and demand for personalized support has created a significant challenge in providing the necessary assistance for students to excel academically and maintain their well-being. Professors, instructors, or teaching assistants (TAs) handle this need for individualized support, resulting in inconsistencies in the treatment of students and their accommodations. This also places an emotional burden on course staff from students' often traumatic experiences. This has the potential to take a significant emotional toll on educators, who often deeply care for their students. Furthermore, in larger courses, students are increasingly expected to advocate for themselves, in addition to coping with their existing challenges.

Courses have adapted to accommodate the growing number of students requiring support. Historically, many courses employ a system of slip days, granting each student a fixed number of days to submit assignments late throughout the course. However, anecdotally this system leads to student decision fatigue and stress—they must weigh using their slip days for a minor task or stockpiling them for a more significant assignment. However, more recently, policies have shifted towards a more individualized and equitable approach to extensions [1, 2]. In this work, we present an early version of the latter; an extensions tool[1] that standardizes and automates the process of requesting extensions, determines who should receive them, sends out emails, and handles extension processing within grading systems. This lightweight tool supports thousands of computer science and data science students each semester.

2 OVERVIEW

This extensions tool (Figure 1) consists of a student-facing Google Form for extension requests, a staff-facing Google Sheet to process requests, and an automation that emails students their extension

[1] https://github.com/berkeley-eecs/extensions

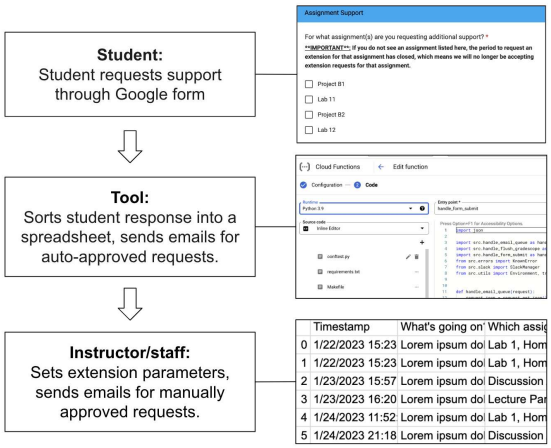


Figure 1: Workflow of the extensions tool backend.

#	Question
1	What is your student ID?
2	What is your name?
3	Are you registered with the campus accessibility program?
4	Which assignment would you like an extension on?
5	How many days would you like an extension for?
6	Why do you need this extension?
7	Are you working with a partner?
8	What is your partner's email and student ID?

Figure 2: Student-facing Google Form questions.

request's status and, if applicable, creates an extension for the corresponding assignment(s) in Gradescope. Gradescope is an assignment grading submission system that manages student deadlines and staff grading.

2.1 Google Form For Extension Requests

To request extensions throughout the semester, students fill out a single Google Form that is regularly updated with the current list of assignments that allow extensions. The form questions (Figure 2) are tailored to fit the needs of the flexible extension policies enacted so far, but are editable and interchangeable as needed per course. These questions have been honed to reduce the load on students and require minimal effort and time to request help without sharing detailed accounts of their personal situations.

2.2 Google Sheet for Extension Approvals

The Google Sheet "backend" is the primary interface for course staff to review extension requests, once Figure 1's workflow is established. In large courses, extension requests are reviewed in part by TAs; lists of TAs with such access are made transparent to students. The backend has five sheets that receive, parse, and interact with data. Data collected from the Google Form is parsed into a "roster" sheet, which records per-student the extension duration per assignment, timestamp, campus accessibility program status (if registered), and the most recent reason for an extension request. Additionally, it indicates the extension request status (automatic, pending approval, or manual) and email status (automatic, pending approval, in queue, or manual). This user-friendly interface streamlines the process for staff, eliminating both the need to field individual extension request emails by student and the overhead of cross-checking campus accessibility program databases.

3 FUTURE CONTRIBUTIONS

The tool currently enables students to easily request extensions and receive prompt responses. The setup involves some computer science knowledge, such as configuring a Google Sheets app script. A walkthrough setup video and a more user-friendly UI that maintains flexibility while minimizing common user errors is in progress. This tool was designed to meet the needs of one course but has expanded to meet demand across a whole department. The tool's adoption across numerous courses at a US public university with a selective CS program has reduced the stigma of seeking help, evidenced by a surge in extension requests with mental health often being cited as the reason why [6]. As the tool scales up to meet greater demand, we anticipate the need for better error support. Our current efforts are focused on an FAQ page to address common issues. In the future, we envision setting up a Google Form or specific email to field requests, potentially for the FAQ page. Since this tool was designed for one course that uses Gradescope for student submissions, it is less efficient with different assignment submission platforms. While the tool can still be used for email automation, updating assignment extensions automatically in a different grading platform is not yet possible, resulting in a time burden on course staff to do so. We next aim to integrate the backend with Canvas. In its current form, the tool is explicitly tailored to the policy of the course for which it was initially created. One side effect is that the tool provides no infrastructure to reject extensions. Courses with different policies may find having to manually inform students of extension rejections a burden. Some courses have attempted to rectify this issue by manually updating the student's request and including a note that the request was not approved. We plan to adapt the tool to allow more flexibility in its email templates, decreasing the burden on courses with different extension policies. Utilizing this tool in as many courses as possible will facilitate the spread of flexible extension policies and support students' success throughout their academic careers. It is our hope that this tool can meet the needs of even larger courses as class sizes continue to scale up.

REFERENCES

[1] Andrew Berns, J Philip East, and J Ben Schafer. 2021. Grading for Equity: A Curriculum Development and Grading Process to Enhance Instruction. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 1351–1351.
[2] Huanyi Chen and Paul AS Ward. 2023. The Value of Time Extensions in Identifying Students Abilities. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. 512–518.
[3] Dan Garcia, Connor McMahon, Yuan Garcia, Matthew West, and Craig Zilles. 2022. Achieving" A's for All (as Time and Interest Allow)". In *Proceedings of the Ninth ACM Conference on Learning@ Scale*. 255–258.
[4] Cormac LaLiberte. 2023. Data science, computer science majors show biggest increase in student enrollment. *The Daily Cardinal* (2023).
[5] Xiaomei Wang, Sudeep Hegde, Changwon Son, Bruce Keller, Alec Smith, and Farzan Sasangohar. 2020. Investigating Mental Health of US College Students During the COVID-19 Pandemic: Cross-Sectional Survey Study. *Journal of Medical Internet Research* (2020).
[6] Jacob Yim, Yuerou Tang, Jordan Schwartz, Madison Bohannan, Dana Benedicto, Charisse Liu, Armando Fox, Lisa Yan, and Narges Norouzi. 2024. Supporting Mastery Learning with Flexible Extensions. (2024).

Barriers to Equitable CS Expansion

Perceptions of Secondary Teachers of Computer Science during COVID-19

Robert Schwarzhaupt
American Institutes for Research
rschwarzhaupt@air.org

Ryan Torbey
American Institutes for Research
rtorbey@air.org

Bridina Lemmer
American Institutes for Research
blemmer@air.org

Joseph P. Wilson
American Institutes for Research
jwilson@air.org

ABSTRACT

To understand secondary teachers of computer science's perspectives on barriers to implementing equitable CS instruction during the COVID-19 pandemic, the research team administered a survey to the CS for All Teachers virtual community of practice in Spring 2021 and examined responses from middle and high school teachers of CS (n = 349). We used the CAPE framework (Capacity, Access, Participation, and Experience) to ground our analysis of survey responses. **Key Finding 1** is that secondary CS expansion is limited due to competing priorities related to college preparation and standardized testing requirements, which do not usually include CS (Capacity; Access). **Key Finding 2** is that student access to broadband at home creates barriers to secondary CS learning (Access; Participation; Experience). These findings reinforce the evidence base that expanding K-12 CS education (before, during, and after COVID-19) faces significant barriers due to gaps and disparities in CS education policies and inequities in students' access to necessary technology.

ACM Reference Format:

Robert Schwarzhaupt, Ryan Torbey, Bridina Lemmer, and Joseph Wilson. 2024. Barriers to Equitable CS Expansion: Perceptions of Secondary Teachers of Computer Science during COVID-19. *In Proceedings of 2024 ACM Technical Symposium on Computer Science (SIGCSE'24), March 20-23, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635529

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

© 2024 Copyright is held by the owner/author(s).

ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635529

1. BACKGROUND

Computer science (CS) is especially relevant to the 21st century, yet there are systemic and structural forces hindering its implementation in American K-12 public schools [1]. In the United States, not all students have access to CS coursework at the high school level, which contrasts with an increasing number of countries worldwide. While CS education has rapidly expanded in the US in the past decade, issues of equity in capacity, access, participation, and experience persist along historic lines of inequity [2, 3]. To further understand barriers to implementing CS instruction, this study solicited the opinions of teachers of CS.

2. SURVEY

In April 2021, the CS for All Virtual Community of Practice research team sent an individualized link via the SurveyMonkey platform to a total of 1,215 community members who logged in to the community website at least once from March 2020 through March 2021 and consented to survey research activities. All research activities were approved by American Institutes for Research's Institutional Review Board. This study descriptively analyzed survey results address two research questions: **RQ1**: What do teachers of CS identify as their top factors limiting their schools' ability to offer or expand CS learning opportunities during the COVID-19 pandemic? **RQ2**: What do teachers of CS identify as barriers to implementing equitable CS learning opportunities during virtual or hybrid instruction?

A total of 539 community members responded to the survey, yielding a response rate of 44% (539/1,215), which is appropriate for data analysis of electronic surveys [4,5]. Survey respondents indicated the grade levels in which they instructed CS from 2020-2021. Those respondents who did not instruct CS in grades 6-12 were excluded from analysis (n = 190), yielding an analytic sample of 349 teachers. Demographics of the analytic sample, including education and experience levels, will be reported on the poster.

2.1 Perceived obstacles to offering or expanding CS learning during COVID-19 (RQ1)

Respondents were asked "To the best of your knowledge over the last year, which of the below reasons is the TOP reason your

school doesn't offer or expand ways to learn computer science?" Respondents were instructed to select one response option from a list. Results are listed in Table 1 below.

Table 1. Frequency of Reported Top Obstacle to Offering or Expanding CS Learning from April 2020-21 (RQ1)

Response Option	n	%
There are too many other courses that students have to take in order to prepare for college	82	25%
We have to devote most of our time to other courses that are related to testing requirements and computer science is not one of them	80	25%
There is not enough demand from students	55	17%
There are no teachers available at my school/in my district with the necessary skills to teach computer science	25	8%
There is not enough money to train or hire a teacher	20	6%
Internet connectivity is poor at my school/in my district or in students' homes	15	5%

*Reponses garnering less than 5% are excluded for space, but will be reported on the poster

2.2 Perceived barriers to equitable CS learning opportunities during virtual or hybrid instruction (RQ2)

Respondents were asked "To what extent do you agree that the following are barriers to providing equitable computer science learning opportunities to all students in your school during virtual or hybrid instruction/support?" for a series of items and were prompted to respond on a 4-point Likert scale: Strongly Disagree (0), Disagree (1), Agree (2), and Strongly Agree (3).

Table 2. Means of Items for Barriers to Equitable CS Learning Opportunities During Hybrid or Virtual Instruction (RQ2)

Item	Mean	SD	n
Student access to reliable broadband internet	2.32	0.82	302
Student familiarity with digital learning tools	2.05	0.83	298
Student access to digital devices	1.95	0.99	304
Teacher familiarity with digital learning tools	1.84	0.94	302
Quality of digital learning materials and programs	1.81	0.93	302
Teacher access to reliable broadband internet	1.63	1.00	305
Teacher access to digital learning tools	1.58	0.95	303

3. FINDINGS

The results of the survey highlight several barriers to expanding equitable CS education. When asked to identify the top reasons their school doesn't offer or expand CS, teachers selected that there were too many other courses that students had to take to prepare for college (25%) and that they had to devote most of their time to other courses that were related to testing requirements (25%). These answers reflect the fact that computer science commonly falls outside the scope of what is considered a priority by schools and districts. As of 2023, no state administers CS standardized testing and, in most states, CS is not a core requirement for college admission, positioning CS outside of college preparation and testing measures that are commonly prioritized by school administration [1]. When teachers were prompted to rate the extent to which they agreed on potential barriers to providing equitable CS learning opportunities to all students in their school during virtual or hybrid instruction/support, the top selection (on average) was 'Student access to reliable broadband internet.' This aligns with other studies, and research by the Kapor Center, which found that barriers related to access to internet at home during the pandemic led to some CS courses being cancelled entirely [6]. Without access to high-speed internet or devices at home needed to complete CS coursework, students are not able to fully participate in CS courses. As with any research, this study is not without its limitations. This is a descriptive study and findings cannot be used to make causal inferences about obstacles in CS education. Moreover, the study's survey sample was selected using non-probabilistic methods, and the sample is not representative of the broader CS teacher population, thus limiting the generalizability of findings.

REFERENCES

[1] Code.org, CSTA, and ECEP Alliance. 2022. 2022 State of Computer Science Education: Understanding Our National Imperative. Retrieved from https://advocacy.code.org/stateofcs

[2] Margolis, J. 2017. Stuck in the Shallow End, Updated Edition: Education, Race, and Computing. MIT Press.

[3] Fletcher, C. L., and Warner, J. R. 2021. CAPE: A Framework for Assessing Equity Throughout the Computer Science Education Ecosystem. Commun. ACM 64, 2, 23-25.

[4] Wu, M. J., Zhao, K., and Fils-Aime, F. 2022. Response Rates of Online Surveys in Published Research: A Meta-Analysis. Comput. Hum. Behav. Rep. 7, 100206.

[5] Nulty, D. D. (2008). The adequacy of response rates to online and paper surveys: what can be done?. Assessment & evaluation in higher education, 33(3), 301-314

[6] Martin, A., Koshy, S., Hinton, L., Scott, A., Twarek, B., and Davis, K. 2021. Teacher Perspectives on COVID-19's Impact on K-12 Computer Science Instruction. Retrieved from: https://www.kaporcenter.org/teacher-perspectives-on-covid-19s-impact-on-k-12-computer-science-instruction.

Supporting Teacher Understanding of Computational Thinking Integration into Early Elementary Curricula

Heather Sherwood†
Education Development Center
HSherwood@edc.org

Alice Kaiser
Education Development Center
AKaiser@edc.org

Camille Ferguson
Education Development Center
CFerguson@edc.org

Anthony Negron
New York Hall of Science
anegron@nysci.org

Ray Ferrer
New York Hall of Science
rferrer@nysci.org

Don Labonte
Participate
don.labonte@participate.com

ABSTRACT

The recent emphasis on computational thinking (CT) integration across disciplines in elementary education nationwide is important to ensure that students are well-equipped to meet the demands of the data-driven workforce. However, this requirement poses major challenges to early elementary teachers who are faced with little direction in terms of how best to implement and assess their young student's application of these new practices.

This project engaged researchers, professional development (PD) providers, and K-2 teachers to develop and design a hybrid PD experience to support teachers in a process of learning about and identifying the use of CT skills and strategies in their students' work. The hybrid PD approach utilized a combination of instructional coaching, both face-to-face and through an online Community of Practice. Researchers used a mixed-methods design to measure teachers' change over time in their understanding of how to identify emergent computational thinking in their students' work process. Data collection methods included teacher focus groups and interviews, classroom observations, and teacher pre- and post-surveys.

Preliminary findings showed an increase in teachers' understanding of CT and confidence in engaging their students in CT-integrated lessons. Teachers reported that the combination of in-person PD and online Community of Practice helped them prepare to bring CT into their classroom and cited placing more of a distinct focus on teaching students the process of CT rather than just using CT vocabulary. Teachers also referenced success in evaluating their students' application of CT skills and concepts by using informal and formative assessment strategies that focus on evidence of student understanding rather than assessing CT vocabulary.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635540

ACM Reference format:
Heather Sherwood, Alice Kaiser, Camille Ferguson, Anthony Negron, Ray Ferrer, and Don Labonte. 2024. Supporting Teacher Understanding of Computational Thinking Integration into Early Elementary Curricula. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGSCE 2024), March 20-23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635540

1 INTRODUCTION

Computational thinking is new to grade K-2 classrooms, having only become a priority following the establishment of the national CS for All initiative in 2016 [1]. Since 2016, a range of new curricula, instructional tools and assessments have been developed to support this new emphasis on both computer science and CT, some of them specifically for early elementary grade teachers and students.

To address the need above, this project sought to produce and investigate a set of sustainable, scalable structures and resources to support high-quality teacher learning about how to recognize, interpret and support students' use of computational thinking to define and solve problems across multiple subject areas for early elementary learners.

2 RELATED WORK

This research builds in part on an inference that early grade computational thinking instruction is similar to early grade mathematics instruction in important ways, and that computational thinking PD should accordingly draw from exemplary models of ambitious early grade mathematics PD. Grade K-2 learners can use distinctive approaches to defining and modeling problems. Rehearsal of these varied approaches to problem-solving during the early grades lays the conceptual and practical groundwork for further disciplinary study in the later grades [2]. While research on early grade CT is not yet wide-ranging, it does suggest that K-2 teachers are facing challenges as they learn the principles of CT but are often not fully prepared to recognize, interpret, and respond to evidence of that type of thinking in their students' work [3; 4].

3 PROJECT DESIGN

This project engaged researchers, professional development providers, and early elementary teachers (K–2) in a collaborative

research and development process to design a scalable professional development (PD) experience for early elementary teachers. Teachers' learning was supported through instructional coaching, both face-to-face during a Summer Bootcamp and through an online community of practice platform. Providers made instructional resources available via this platform, which allowed teachers to extend their learning beyond the in-person PD.

The research questions that guided this work included: 1. What kind of PD and guidance do teachers need to identify and support emergent computational thinking development in young students' language and work process? and 2. What kind of PD and guidance do teachers need to identify emergent computational thinking development in young students' work products?

3.1 Methods

The research used a mixed-methods design to examine teachers' change over time in their understanding of how to identify emergent computational thinking in their students' work process. Researchers analyzed teacher interviews and focus groups, observations, and teacher pre- and post-surveys.

3.2 Participants

22 K-2nd grade teachers from seven New York City schools participated in the face-to-face Summer PD training and then engaged in the Community of Practice (CoP) using an online platform throughout the 2022-2023 school year. Nine teachers from three schools participated in additional research activities, including the teacher interviews and focus groups, teacher surveys, and classroom observations.

4 RESULTS

4.1 Supporting Teacher Understanding of CT Development

One goal of this project was to learn what kind of professional development and support teachers' need when learning how to engage early elementary learners in computational thinking. To understand this, the research team asked teachers to reflect on the different aspects of the professional development program they engaged in over the project period, including attending the in-person Summer PD, engaging in the CoP, self-directed use of the online platform and resources, and any additional outside resources they sought after to deepen their CT knowledge.

Preliminary data analysis of teacher surveys, interview and focus group data showed an increase in teachers' understanding of CT and confidence in engaging their students in CT-integrated lessons. Teachers described how the in-person PD helped them to focus on the process of using CT, instead of just teaching the vocabulary. In addition, teachers appreciated the hands-on nature of the Summer PD, which allowed for opportunities to watch the PD providers model lesson activities with children during work periods. Teachers emphasized how instrumental this live-action modeling proved to be in preparing to bring CT into their own classroom.

Teachers also stated that their primary goals for using the PD resources and CoP on the platform was to refresh their memory of CT concepts presented in the Summer PD and to look at other teachers' lesson plans for ideas. Additionally, teachers found value in being a part of the CoP and highlighted being able to share ideas with colleagues for feedback and maintain connections with colleagues outside of their school as two main benefits.

However, when asked about platform usage and their engagement in the CoP, teachers consistently stated that they did not visit the platform often, did not use resources on the platform in depth and did not use the platform for planning or communicating with teachers from their own school. They stated that they would visit the platform when there was a specific purpose (such as sharing their lesson) or when they were prompted to by reminder emails from the PD team.

4.2 Supporting Teachers in Identify CT in Student Work Products

Another main goal of the project was to learn how teachers are identifying CT in young learners' work process. In order to evaluate student learning and specifically the CT strategies applied, teachers described using informal and formative assessment practices, such as observing students, listening to peer discussions, and asking students direct questions. When asked how teachers determined which strategies to employ when looking for evidence of students' application of CT in classwork, they cited drawing on activities and examples that they were modeled by PD providers during the Summer PD, drawing on their own Pedagogical Content Knowledge and prior experiences as a teacher.

Furthermore, teachers referenced looking for evidence of students using CT practices, such as breaking problems down, and by looking at the process of problem-solving students were engaging in, rather than listening for explicit CT vocabulary. Teachers added rationale to their decisions for focusing on the process instead of CT vocabulary by stating that it is challenging to focus on CT vocabulary with young learners, especially those that are Emergent Bilingual or English Language Learners, as they often have limited understanding of English and CT vocabulary can be too advanced or complicated for them.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2101547. We are grateful to the district staff, school administrators, teachers, and students involved in this work, without whom this research would not be possible.

REFERENCES

[1] DeLyser, L. A., Wright, L., Wortel-London, S., & Bora, A. (2020). Evaluating a systems approach to district CS education implementation. In *Proceedings of the 51st ACM technical symposium on computer science education* (pp. 1120-1126).

[2] Rich, K. M., Yadav, A., & Schwarz, C. V. (2019). Computational thinking, mathematics, and science: Elementary teachers' perspectives on integration. *Journal of Technology and Teacher Education, 27*(2), 165–205. https://www.learntechlib.org/primary/p/207487/.

[3] Grover, S., Jackiw, N., & Lundh, P. (2019). Concepts before coding: Non-programming interactives to advance learning of introductory programming concepts in middle school. *Computer Science Education*, *29*(2-3), 106-135.

[4] Yan, W., Liu, R., Israel, M., Sherwood, H., Fancsali, C., & Pierce, M. (2020). School-wide integration of computational thinking into elementary schools: A cross-case study. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 1325-1325).

CT Readiness for All: Studying a Framework for Supporting Schoolwide Computational Thinking Integration Across Elementary Curricula

Heather Sherwood
Education Development Center
New York, NY USA
hsherwood@edc.org

Babette Moeller
Education Development Center
New York, NY USA
bmoeller@edc.org

Cheri Fancsali
The Research Alliance
for NYC Schools
New York, NY USA
cheri.fancsali@nyu.edu

ABSTRACT

Schools throughout the United States are engaging in efforts to integrate computational thinking (CT) across various elementary curricula. However, there is very little guidance for effective approaches to integrating CT consistently and cohesively school wide. CT Readiness for All is a two-year research project that is investigating a CT framework and self-assessment tool developed to articulate indicators associated with successful school-wide integration across elementary curricula. Data sources include focus group interviews and surveys with teachers. Although the project is still in progress, early analysis have resulted in three key findings: (a) students were able to make cross-curricular connections using CT as a problem-solving process; (b) finding time within the school day to focus on CT is challenging; and (c) administrators need to take an active role in setting the vision and definition of CT to support school-wide CT efforts.

ACM Reference format:
Heather Sherwood, Babette Moeller, Cheri Fancsali, 2024. CT Readiness for All: Studying a Framework for Supporting Schoolwide Computational Thinking Integration Across Elementary Curricula. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635533

1 INTRODUCTION

CT is conceptualized as a set of concepts, skills, and practices for problem solving that draws from computer science (CS), involves computational devices and is applicable to many different disciplines in education [1; 2]. Research to date indicates that the application of CT improves students' ability with problem-solving and understanding of concepts in core-content areas [3; 4]. However, CT integration across elementary curricula requires a coordinated effort within a school to help develop students' skills outside of computer science (CS) classes. Additionally, schools need to engage in a continuous and intentional schoolwide effort to develop CT competencies across subject areas.

This project is conducting research to examine how schoolwide integration of CT in grades PreK – 5 contributes to developing the interest in and preparation for CS for students from diverse backgrounds. The goal of this project is to field-test the CT framework and self-assessment tool to investigate their promise for helping schools create school-wide CT integration plans, providing equitable access to CT education for underrepresented students in CS.

2 PROJECT DESCRIPTION

While there are many factors that influence the quality of CT integration, the CT framework provides educators with a tool to identify the specific variables that are likely to have an impact on integration efforts. The CT framework outlines hypothesized determinants of successful integration for educators to evaluate their readiness for, strategically plan for, and continuously evaluate progress towards school-wide CT integration by identifying six focus areas of successful integration: (1) Teacher Knowledge, Pedagogical Content Knowledge, and Facility with Tools to Support Student Learning of CT and Core Subjects; (2) Teacher Supports; (3) Curriculum Features and Lesson Planning; (4) CT Assessment; (5) Student Impact; and (6) Families.

This study is investigating how elementary schools utilized the CT framework as a tool for integrating CT into their school settings, and the extent to which the implementation of the CT integration planning process results in students' increased access to and participation in high-quality CT instruction. During the first year of the study, we worked with six teachers from one school located in New York City. These six teachers included one Technology/Media teacher, two Pre-K through 5th grade Art teachers, one K through 5th grade Science teacher, one 3rd through 5th grade Social Studies teacher, and one K through 5th grade Gym teacher. Because these teachers each saw multiple classes, they worked with students with mixed abilities including English Language Learners and students with disabilities.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635533

4 RESULTS

This project was designed to work with schools over an extended period to document the way in which the framework supports them with scaling and sustaining their CT integration efforts. Although the project is still ongoing, three preliminary findings were identified in early analysis of data.

4.1 *Students made cross-curricular connections of CT concepts in their work.* After engaging with the framework as a structure for developing a cohesive definition of CT and identifying the specific CT concepts and practices the school will focus on, the participating teachers noted hearing students in different content-area classes make cross-curricular references to how they were applying CT as a problem-solving strategy in their other classes. For example, one teacher in a focus group said, *"Students would actually go "oh I remember this in art, I remember this in technology" and the terms are starting to become more organic and understood, so I feel like that was a really great approach to introduce this as a tool you use anywhere, it's problem solving."*

The participating teachers noted that it was necessary for them to develop a shared understanding of the role CT will play in their curriculum and that in order to successfully integrate CT systematically across all classes, they needed to agree upon the specific CT concepts and approaches that will be the foundation for work. In early PD sessions, teachers revealed that they did not have a shared definition of CT, nor did they focus on the same CT concepts and practices, and this led to them all feeling as if they were working independently and not towards the same goal. They also felt that while there might be teachers who introduced CT as a part of their curriculum before participating in this project, it was sporadic and not in a unified way. By the end of the first year, the teachers reported that a successful outcome of their work was that they solidified a definition of CT and used the same CT vocabulary consistently across their classes.

4.2 *Finding time to integrate CT within content was challenging.* Teachers identified lack of time within the school day as a major challenge when undertaking CT integration. This sentiment was expressed both by teachers in the focus group and in the survey. In the survey, all five teachers stated that lack of time was a moderate or great challenge to integration.

4.3 *Administrators need to take an active role setting the vision and definition of CT to support school-wide CT efforts.* Teachers also identified the need for additional support from administrators as a key component to furthering their CT integration efforts. In the focus group, teachers stated that they felt as though they made significant process in the beginning of the school year but got to a place where they needed administrative input as to what their definition of CT is as well as the main CT concepts and practices administration wanted to see embedded into lesson plans to be able to move forward.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2219314. We are grateful to the district staff, school administrators, teachers, and students involved in this work, without whom this research would not be possible.

REFERENCES

[1] Wing, J.M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

[2] Yadav, A, Hong, H., & Stephenson, C. TechTrends (2016) 60: 565. https://doi.org/10.1007/s115280160087-7

[3] Calao, L.A., Moreno-León, J., Correa, H.E., & Robles, G. (2015) Developing mathematical thinking with scratch. In C. Gráinne K. Tomaz, R. Christoph, K. Johannes, & E. Lavoué, (Eds., *Design for teaching and learning in a networked world* (pp. 17-27). Springer, Cha.

[4] Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field.

Novices' Perceptions of Web-Search and AI for Programming

James Skripchuk*
North Carolina State University
Raleigh, North Carolina, USA
jmskripc@ncsu.edu

John Bacher
North Carolina State University
Raleigh, North Carolina, USA
jtbacher@ncsu.edu

Yang Shi
North Carolina State University
Raleigh, North Carolina, USA
yshi26@ncsu.edu

Keith Tran
North Carolina State University
Raleigh, North Carolina, USA
ktran24@ncsu.edu

Thomas Price
North Carolina State University
Raleigh, North Carolina, USA
twprice@ncsu.edu

ABSTRACT

External help resources are frequently used by novice programmers solving classwork in undergraduate computing courses. Traditionally, these tools consisted of web-based resources such as tutorial websites and Q&A forums. With the rise of AI code-generation and explanation tools, understanding how students use external resources and their roles in classroom have become especially relevant. Despite this, little research has directly investigated the extent to which students *intend* to use these tools and what factors influence their beliefs. It is unknown when students think it is appropriate to use these tools and what features they find valuable. Understanding these beliefs would allow instructors and researchers to better focus their efforts on what aspects of pedagogy and tool usage should be addressed. We administered a pilot vignette-style survey to introductory programming classes at an R1 University ($n = 45$), giving students scenarios of external resource usage while questioning their attitudes, subjective norms, and their perceived behavioral control on using these external resources. We share preliminary findings on free response data, showcasing the variety of beliefs and opinions that novice programming students have on when and how much external resource usage is acceptable in the classroom. Some students felt that AI tools can provide more exact solutions than searching for help online, but also expressed that this exactness could be detrimental to their learning. Others expressed awareness that professionals use these resources, and expressed a desire to learn how to use them in a way to help their educational and career goals.

ACM Reference Format:
James Skripchuk, John Bacher, Yang Shi, Keith Tran, and Thomas Price. 2024. Novices' Perceptions of Web-Search and AI for Programming. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635545

*This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-2137100.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635545

1 MOTIVATION AND PROCEDURE

Programmers often use external resources to both learn and seek help online while programming. This can range from **web-based** help such as documentation and crowd-sourced Q&A forums, to **AI-based** code generation and explanation tools such as ChatGPT. Being able to effectively navigate, judge, and utilize a variety of scattered online resources is seen as a critical skill for professional programmers. With increasing access to these resources, particularly AI-based tools, instructors and researchers have begun to reevaluate external resources' place in the classroom. However, little research has directly investigated the extent to which students *intend* to use these resources in the classroom in the first place, and what factors might lead to these decisions.

It has been shown the novices make frequent use of web-based resources in the classroom, even if they find them hard to navigate. Students tend to move quickly and engage in "trial-and-error" type search behaviors [2]. AI tools have shown to improve programming efficiency in professionals, sometimes at the cost of understanding. Novices' face similar challenges, with some expressing mixed opinions of how AI-tools can influence their learning [1]. However none of this research specifically captures students' thoughts of these resources in their learning process.

Procedure: The original survey is modeled off of the Theory of Planned Behavior, which posits that three interconnected factors influence an individual's intention to perform a behavior: **Attitude**, **Subjective Norm (SN)**, and **Perceived Behavioral Control (PBC)**. Due to the scale of the data, in this poster we provide only a subset of our results based on thematic analysis of the free response data, using these three factors as a guide.

We surveyed three classes at an R1 (Carnegie Classification) University from a suburban area on the east-coast of the United States. Participation was voluntary. We surveyed two CS1 classes for non-majors, taught in Python (24/60) and MATLAB (14/260) respectively, and one CS2 class for majors taught in Java (7/150) for a total of 45 respondents. The university had restrictions on external resources: use of external resources other than approved documentation sites would be count as an academic integrity violation. Participants were entered into a random drawing to win one of multiple 25$ Amazon Gift Cards. Out of the 45 participants: 35 were male, 8 were female, and 2 preferred not to answer. 24 were White, 4 were Asian, 4 were South Asian, 3 were Hispanic/Latinx, 7 were two or more racial identities, and 3 preferred not to answer. 21 of the respondents indicated that they took at least one programming

course in High School. 23 students have used an AI tool at least once for programming, 21 students heard of AI tools but never used them for programming, while 1 had not heard of any AI tools.

We utilized thematic analysis in order to identify common perceptions and themes in the free response data. Four researchers gathered in the same room and initially looked over the 90 free response question responses, wrote down high-level notes on the perceptions that they saw, and then discussed these behaviors. They then refined the initial list of perceptions and opinions that they noticed into concrete codes. The four researchers then went through all of the data again, coming to a consensus on the codes for each response, before merging codes into themes.

2 RESULTS

We found that students had a wide variety of opinions about when, where, and how much external resource usage was acceptable in a classroom context. One of the most important criteria participants used to decide whether to use a resource is how applicable the examples they find are to the problem they are solving, i.e. how "specific" and "exact" they are. Some students stated that external resources in general were unable to provide relevant help:

- "*I don't feel like getting outside code would be useful because I need code for a specific assignment*" (P18, CS1)
- "*I think that it would be easier to look through notes than to look to the Internet for more code for debugging, because the code it might provide won't be specific enough for the assignment you have*" (P27, CS2).

Some said that the primary value of web-search is *not* to provide exact solutions, but instead to serve as a guidance or reference. Some saw this lack of specificity as a positive:

- "*Internet resources are much more unlikely to find a solution to your exact problem, and will simply give a direction to look at*" (P5, CS1)
- "*Usually when you look for certain code [on the internet] it will list different ways to use certain material used in class, but you'll never find the exact answers*" (P18, CS1).

Students compare web-search to AIs ability to create bespoke solutions, which some consider closer to cheating. Even though AI is prone to "hallucinations", it has become increasingly competent at solving introductory programming problems:

- "*AI can give you code that exactly fits your situation*" (P6, CS1)
- "*[AI is] definitely closer to cheating*" (P44, CS1)
- "*[AI] doesn't really teach you anything*" (P5, CS1)
- "*[AI is] a good option if just one function is not working, almost like a debugger, but it is not something that should be used for the whole project*" (P45, CS1)

Students found these help-resources especially useful for starting points and idea generation. Some also found AI especially useful for debugging, even though research has shown that AI-based help can be flawed:

- "*AI tools and the Internet only give me a snippet into what type of code I should use, I feel like they help me when I'm stuck and don't know where to turn*" (P7, CS1)
- "*Debugging with AI can be useful, and it also can help give new suggestions; just like writing English, it's great to have new eyes read your work*" (P41, CS1)
- "*AI can cater to your specific needs for an assignment. You can also ask it questions in real time, making it easier to understand what it is doing and why*" (P17, CS1)
- "*It gives you an in-detail explanation of what is wrong, however, putting your whole code in there is not helpful, you need to know what line is not working*" (P45, CS1)
- "*I have found that GPT can be flawed in its responses, which leads me to use it as a resource to better my understanding or help debugging of an assignment rather than just copying the code it spits out*" (P42, CS1)

Students have mixed responses on how using these resources influence their learning, some saying that using external resources are harmful in general, while others saying that they can be beneficial only if one puts in the effort to understand the solution - a point of entry for instructors to teach robust help-seeking skills:

- "*using these [resources] would lead to less learning*" (P33, CS2)
- "*there is a difference between copying code directly and taking the time to learn something based on something found online*" (P36, CS2)
- "*it can hurt when you don't understand the fix and just plug it in*" (P5, CS1)

Some students were wary about getting caught, but were unaware of methods to detect the use of these resources:

- "*there may be some way to stop people from using the Internet and AI to help them with their code, but I do not know if those exist*" (P16, CS1)

Finally, some students expressed some knowledge that these tools are sometimes used by professionals and expressed a disconnect between their current education and their perceived postgraduate goals, showing an opportunity for educators to be more clear to students in their pedagogical goals:

- "*in a real-world scenario these would be resources that would assist me and I would have access to*" (P13, CS1)
- "*We will be using these things in the work force, so why not learn how to use them now*" (P14, CS1)
- "*[the use of these tools] is pretty realistic in the real world. Rarely do you have to do something completely from scratch*" (P5, CS1)
- "*cheating isn't real and 'not allowed to show others your code' is a dumb rule that nobody in real life adheres to*" (P22, CS2)
- "*I have talked to many Software engineers who regularly use the Internet in their careers to look up how to solve a problem, but do not copy-paste code*" (P39, CS1)

REFERENCES

[1] James Prather, Brent N. Reeves, Paul Denny, Brett A. Becker, Juho Leinonen, Andrew Luxton-Reilly, Garrett Powell, James Finnie-Ansley, and Eddie Antonio Santos. 2023. "It's Weird That it Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. *ACM Trans. Comput.-Hum. Interact.* 31, 1, Article 4 (nov 2023), 31 pages. https://doi.org/10.1145/3617367

[2] David Wong-Aitken, Diana Cukierman, and Parmit K. Chilana. 2022. "It Depends on Whether or Not I'm Lucky": How Students in an Introductory Programming Course Discover, Select, and Assess the Utility of Web-Based Resources. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE* 1 (2022), 512–518. https://doi.org/10.1145/3502718.3524751

The Landscape of Disability-Related K-12 Computing Education Research

Julie M. Smith
CSEdResearch.org
Peoria, US
julie@csedresearch.org

Monica M. McGill
CSEdResearch.org
Peoria, US
monica@csedresearch.org

ABSTRACT

Correcting the under-representation of people with disabilities in computing and computing-related careers will require a research base that enables equitable computing education (CE) for all students. For this study, we analyzed research papers in order to better understand the landscape of CE research as it relates to students with disabilities. Of the 771 papers reviewed from 12 major CE venues over the last decade, very few (n = 14) specify student participants' disability status; this <3% rate is under-representative, given that about 15% of students in the US have a disability. Studies that do specify disability status tend to involve students who are in the middle grades, in the United States, and/or are participating in elective activities. Demographic factors (e.g., gender and race) are specified about half of the time and analyzed about half as often as they are specified. A small cadre of researchers (51 from 22 different institutions) are responsible for the studies specifying disability status. Studies – whether they specify disability status or not – most commonly use block-based programming, which may present accessibility issues for some students. Advancing equity for all students will necessitate much more research on the computing experiences of students with disabilities, including reporting factors such as demographic data that make it easier to determine whether and how the research applies in a given context. More research focused on accessible tools and languages for learning computing is also needed to further understand promising practices for teaching students with disabilities.

CCS CONCEPTS

• **Social and professional topics → K-12 education**; **People with disabilities**.

KEYWORDS

computing education research, disability, equity, accessibility

ACM Reference Format:
Julie M. Smith and Monica M. McGill. 2024. The Landscape of Disability-Related K-12 Computing Education Research. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635552

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635552

1 INTRODUCTION AND BACKGROUND

There is a moral imperative to ensure that students with disabilities, who constitute more than one out of ten K-12 students in the United States [4], have equitable access to computing education (CE). But, currently, people with disabilities are underrepresented in computing [3] and emerging technologies – including those used in education, criminal justice, health care, and workplaces – often discriminate against people with disabilities [2].

Remedying these inequities is a multi-faceted process, one part of which is ensuring that the CE research base adequately represents the experiences of students with disabilities. Historically, CE research has not often specified demographic information about study participants [5], making it difficult to determine how and whether findings apply to all student groups; this phenomenon is particularly true for students with disabilities since even when equity issues are addressed, disability is rarely one of the dimensions of diversity taken into consideration. Researchers sometimes do not collect data on student disability because it is difficult to collect and interpret [1]; thus, there is little data on CE students with disabilities [8].

This project seeks to answer the question, *What gaps in computing education research exist that may be contributors to the lack of equity for students with disabilities?*A better understanding of these gaps will provide the CE research community with insight into what future research is needed.

Other barriers to the full participation of disabled students in computing includes false perceptions about their abilities, gaps in teachers' knowledge about how to support students, lack of accessible curricular materials, and a lack of data about student needs [7]. Previous research has found that accessibility problems in K-12 CE exist in five different areas: computer input (e.g., requiring students to use a mouse), computer output (e.g., creating charts), infrastructure (i.e., back-end technologies), curriculum accessibility, and inclusion (specifically, the inclusion of people with disabilities in CE materials) [6].

2 METHODOLOGY

A publicly available dataset of K-12 CE research exists that vets and then curates relevant articles from over a dozen venues that publish CE research. The inclusion criteria require that articles (1) describe or assess a computing activity, (2) focus on K-12 students and/or their instructors, and (3) focus on an activity whose goal is teaching a computing or computational thinking concept. Using the manually curated data for each of the 1200 papers published between 2013 and 2022 inclusive (consisting of over 40 variables, including participant demographics), we analyzed the 771 research

studies on disability-related factors. We considered factors associated with quality research (e.g., specifying a research question), with contextualizing findings (e.g., students' racial background), and with other aspects of studies that might impinge on their ability to reflect the experiences of students with disabilities (e.g., use of potentially inaccessible programming languages and tools).

3 RESULTS

Even in contexts where it was applicable (i.e., there were student study participants), students with disabilities are rarely specified in studies and, even more concerning, there are even fewer studies that study interventions and their impacts specifically on students with disabilities, with only 2.6% of studies (14 articles) in the entire data set of 771 papers specifying student disability status. This percentage has been increasing slightly over the past decade. A total of 51 authors from 22 different institutions contributed to these studies.

There was a roughly even ratio of quantitative (5 studies), qualitative (5 studies), and mixed methods (4 studies) research. Table 1 shows the percent of studies specifying and analyzing various characteristics where applicable to the study.

Factor	Percent	Notes
Study Characteristics		
Was activity required?	42%	4:1 ratio of elective:required
Research question(s)	86%	
What concepts taught?	82%	
Participant Count	75%	Most common: 26 - 50 students (3 papers) and 101 -500 students (3 papers)
Student Characteristics		
Geographic location	69%	67% in the US 33% outside of the US
Socioeconomic status	54%	29% analyzed SES
Gender	54%	36% analyzed gender
Race	43%	20% analyzed race or ethnicity
Ethnicity	23%	20% analyzed race or ethnicity
Age	46%	Most common: 10 - 12 years old
Grade level	76%	Most common: 6th and 7th grades
Prior computing experience	31%	

Table 1: Percent of studies specifying various characteristics (where applicable).

Of those specifying whether an activity was required, four studies involved elective activities and just one a required activity. Because some student populations (e.g., girls) are less likely to know about and/or to participate in elective computing activities [10], the prevalence of elective activities may present a barrier to equitable CE research.

The programming language used in studies may present another barrier. For the articles that specified student disability, the most commonly referenced language was Scratch (in 6 of the 14, or 43%, of the studies). Across all research studies (n = 771) in the data set, Scratch was also the most common language, and four of the most cited papers in this set concern block-based programming. But visually-based languages are not easily accessible for users with cognitive impairments [11] or, especially, limited vision [9].

4 DISCUSSION AND CONCLUSION

The most significant finding of this study is that CE research that considers student disability status is quite rare; more research is clearly needed. Of the research that does exist, some factors (e.g., specification of research question(s) and what topics were taught) are frequently specified. Others, however, such as whether an activity was required or student demographic characteristics, are included much less often. This lack of specificity can make it more difficult to determine whether and how research applies in various contexts. Further, research tends to focus on students in the middle grades, implying a need for more studies with elementary or high school students. Finally, the prevalence of block-based programming languages in CE research may present an obstacle to studying the experience of students with disabilities.

ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. National Science Foundation under Grant Nos. 1625005, 1625335, 1745199, 1757402, 1933671, and 2122212. We also recognize Amazon Future Engineer for their support of the Resource Center.

REFERENCES

[1] Brianna Blaser and Richard E. Ladner. 2020. Why is data on disability so hard to collect and understand?. In *2020 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, Vol. 1. IEEE, 1–8. https://ieeexplore.ieee.org/abstract/document/9272466/

[2] Lydia XZ Brown, Ridhi Shetty, Matthew U. Scherer, and Andrew Crawford. 2022. Ableism and disability discrimination in new surveillance technologies. *Center for Democracy & Technology* (2022).

[3] Sheryl E. Burgstahler and Richard E. Ladner. 2007. Increasing the participation of people with disabilities in computing fields. *Computer* 40, 5 (2007), 94–97. https://ieeexplore.ieee.org/abstract/document/4198258/ Publisher: IEEE.

[4] National Center for Education Statistics. 2023. Students With Disabilities. https://nces.ed.gov/programs/coe/indicator/cgg

[5] Mark Guzdial and Benedict du Boulay. 2019. The history of computing. *The Cambridge handbook of computing education research* 11 (2019). Publisher: Cambridge University Press.

[6] Earl Huff, Varsha Koushik, Richard Ladner, Stephanie Ludi, Lauren Milne, Aboubakar Mountapmbeme, Margaret Perkoff, and Andreas Stefik. 2020. *Accessible Tools and Curricula for K-12 Computer Science Education.* Technical Report.

[7] Mina Huh and Gesu India. [n. d.]. Reimagining Accessibility and inclusion in K-12 CS Education through curriculum and professional development. ([n. d.]). https://www.microsoft.com/en-us/research/uploads/prod/2021/01/Reimagining-Accessibility-and-Inclusion-in-K12_Making-Higher-Ed-in-CS-Accessible-Group-B.pdf

[8] R. Ladner. 2020. Expanding the Pipeline: The Status of Persons with Disabilities in the Computer Science Pipeline. *Computing Research Association-Widening Participation. Retrieved June* 19 (2020), 2022.

[9] Lauren R. Milne. 2017. Blocks4All: making block programming languages accessible for blind children. *ACM SIGACCESS Accessibility and Computing* 117 (Feb. 2017), 26–29. https://doi.org/10.1145/3051519.3051525

[10] Jennifer Wang and Sepehr Hejazi Moghadam. 2017. Diversity Barriers in K-12 Computer Science Education: Structural and Social. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education.* ACM, Seattle Washington USA, 615–620. https://doi.org/10.1145/3017680.3017734

[11] Misbahu S. Zubair, David Brown, Thomas Hughes-Roberts, and Matthew Bates. 2018. Evaluating the Accessibility of Scratch for Children with Cognitive Impairments. In *Universal Access in Human-Computer Interaction. Methods, Technologies, and Users*, Margherita Antona and Constantine Stephanidis (Eds.). Vol. 10907. Springer International Publishing, Cham, 660–676. https://doi.org/10.1007/978-3-319-92049-8_49 Series Title: Lecture Notes in Computer Science.

Reimagining CS Courses for High School Students

Julie M. Smith
CSEdResearch.org
Peoria, US
julie@csedresearch.org

Bryan Twarek
CSTA
Chicago, US
bryan.twarek@csteachers.org

Monica M. McGill
CSEdResearch.org
Peoria, US
monica@csedresearch.org

ABSTRACT

Traditionally, computer science (CS) in the United States has been an elective subject at the high school level. In recent years, however, some school systems have created a CS graduation requirement. Designing a required CS course that meets the needs of anticipated future advancements in the field necessitates exploring the research question, *What computing content do high school teachers, college instructors, and computing industry professionals prioritize in a required computer science course for high school students?*To better understand what these different groups perceive to be the essential content of a foundational high school CS course, we conducted a series of focus groups. These focus groups explored participants' (n = 21) thinking about what content would be most important to prioritize in a required high school CS course. Transcripts of the focus groups were abductively coded and then analyzed to determine what CS content priorities were identified and what disagreements about priorities exist.

We found that participants (1) emphasized CS knowledge and skills, with minimal reference to dispositions, (2) prioritized content similar to that found in current CS standards, (3) developed broad, high-level descriptions of content, (4) identified contextually relevant factors, (5) foregrounded AI both a tool and as a subdomain of CS, and (6) emphasized computational thinking. These findings can inform further research on the design and implementation of a required high school CS course designed to meet the needs of the future as well as to support revisions of CS standards for high school students.

CCS CONCEPTS

• **Social and professional topics → K-12 education**.

KEYWORDS

high school, focus groups, standards, curriculum

ACM Reference Format:
Julie M. Smith, Bryan Twarek, and Monica M. McGill. 2024. Reimagining CS Courses for High School Students. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635554

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635554

1 INTRODUCTION AND BACKGROUND

In recent years, eight states have made CS a graduation requirement [4], a policy now recommended by groups including Code.org [3]. The development of required CS courses represents an opportunity and a challenge: an opportunity to expose all students to important computing concepts as well as a challenge to create a course that meets the needs of all students while adequately introducing a rapidly expanding and quickly changing discipline that has enormous societal implications.

2 METHODOLOGY

Participants In the fall of 2023, we distributed a call for participation in a project with the purpose of determining what CS content is essential for all high school graduates. Nearly 300 respondents expressed interest in participating in the project in some way. Of those applicants, we chose approximately thirty to participate in one of three focus groups, one each for those whose primary role is in high school CS, higher education CS, or industry CS. We chose participants via a semi-random process that prioritized diversity across a variety of factors, including geographic (i.e., US region as well as urban/suburban/rural), expertise, experience, demographic, and institution type. The project was classified as exempt from IRB review. The focus groups were held virtually, with each group meeting three times, once per week for one hour.

Data collection While the majority of the meetings consisted of discussion between the participants (prompted by the facilitator as needed), some activities were used to seed the discussions. For example, participants were asked to imagine that they were given the task of designing from the ground up a required CS class, with no restrictions on their courses. Their descriptions were shared and then used as the basis for discussion. Meetings were recorded and then transcribed.

Data Analysis The transcriptions were abductively coded [8], using Thompson's eight-step approach for abductive thematic analysis [7]. The focus of the analysis was to better understand what CS content focus group participants would prioritize in a required high school CS course and what disagreements exist about that prioritization.

3 RESULTS AND IMPLICATIONS

Six major themes were identified in the transcripts; those themes, along with their implications, are explored in this section.

Knowledge and skills, not dispositions. The slide used to introduce the exercise where participants created seven statements to describe a proposed foundational CS class included the words *knowledge*, *skills*, and *dispositions*, meant to serve as a reminder to participants to consider dispositions [2]. However, there were no references to dispositions in the statements developed by almost all of the

participants; instead, they focused on knowledge and skills. The only exception to this trend was one item suggested by one participant who mentioned that students should be able to see themselves continuing their study of computing. Far more commonly mentioned were aspects of computational thinking (e.g., understand the process of abstraction), basic programming concepts (e.g., use functions), and social implications of computing (e.g., identifying ethical issues). Future research should explore the reason for this lack of emphasis on dispositions (e.g., did participants deem them a lower priority, too difficult to assess, hard to articulate, etc.).

Similarities to current CS standards. While there was some variance in the content that participants prioritized, there was extensive overlap between the majority of their suggestions. That is, a large majority of items mentioned by participants as priorities for a high school CS course have a parallel in, for example, the CSTA standards for ninth and tenth graders [1]. Future research might explore the extent to which this overlap reflects the adequacy of current standards to meet future needs, the participants' sense of being circumscribed by current standards, and/or other factors.

Broad, high-level descriptions. The majority of priorities identified by participants were formulated at a high level, such as "manage and manipulate data structures" and "design and implement algorithms." It is understandable that, asked to create a very small number of course descriptors, participants would have created broad statements, but it is also the case that consistent implementation of standards requires much more specificity. For example, which data structures should high school students know how to manipulate, and what kinds of manipulation should they be able to perform? There is also a tension between the desire for flexibility (e.g., across programming languages) created by broader standards and the desire for specificity (i.e., to establish uniform expectations at the college level for what high school graduates should know).

Contextually relevant factors. Participants' priorities were informed by their particular circumstances. For example, one rural high school CS instructor was unable to assign homework because many students live in areas that do not have broadband internet; another instructor was in the process of working with a local college to offer dual credit CS courses. These and similar circumstances informed their perspectives about what priorities, activities, and pedagogies were reasonable in CS courses.

AI as tool and as subdomain. Participants' comments reflected current high levels of interest in generative AI [5, 6] and frequently mentioned how emerging AI technologies could impact CS instruction at several levels: as a tool computing students might use (e.g., for assistance writing code), as an emerging area of computing that students will need to understand conceptually and technically, as a driver of a future decrease in and/or change in the nature of computing jobs, and as a source of new ethical issues. The rapidly changing nature of AI technologies as well as the uncertainty around their impacts makes the topic of AI in computing education both important and elusive.

Computational thinking. Computational thinking was a prominent theme in participants' articulation of priorities, manifested in two main ways: first, specific reference to computational thinking and its constituents (e.g., abstraction, decomposition) were mentioned as key priorities for high school CS. Second, discussions of the future of CS – including industry needs – frequently emphasized that rapidly changing technologies necessitate practitioners who understand CS at a conceptual level and whose training extends beyond the use of currently common tools, enabling them to react nimbly to changing technologies over the course of their careers.

Areas of disagreement. There were some disagreements among participants. For example, high school CS instructors engaged in a thoughtful discussion of whether topics associated with digital literacy and digital citizenship belong in CS courses or in other courses. Similarly, those with industry experience debated whether tools such as version control systems (e.g., Git and GitHub) should be introduced in high school and, more generally, how closely high school CS courses should align with industry needs as opposed to providing a more general CS foundation.

4 CONCLUSIONS

This project sought to explore what CS content is foundational. There was broad agreement about topics (e.g., computational thinking), with some disagreement at the margins (e.g., importance of specific tools). Overall, the findings indicate that it is possible to propose, at least broadly, the content of a required high school CS class. Future work will seek additional input from a variety of relevant parties to further delineate and prioritize content and determine how to meet the challenge of moving beyond the status quo in the face of rapidly emerging technologies.

ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. National Science Foundation under Grant No. 2311746.

REFERENCES

[1] Computer Science Teachers Association. [n. d.]. K–12 Standards. https://csteachers.org/k12standards/

[2] Guy Claxton. 2009. Cultivating positive learning dispositions. In *Educational Theories, Cultures and Learning*. Routledge.

[3] Code.org. 2023. Ten Policy Ideas to Make Computer Science Foundational to K–12 Education.

[4] Code.org, CSTA, and ECEP Alliance. 2022. *2022 State of Computer Science Education: Understanding Our National Imperative.* Technical Report. https://advocacy.code.org/stateofcs

[5] Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutcheme, Lilja Kujanpää, and Juha Sorva. 2023. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1 (ICER '23, Vol. 1).* Association for Computing Machinery, New York, NY, USA, 93–105. https://doi.org/10.1145/3568813.3600139

[6] Sam Lau and Philip Guo. 2023. From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1 (ICER '23, Vol. 1).* Association for Computing Machinery, New York, NY, USA, 106–121. https://doi.org/10.1145/3568813.3600138

[7] Jamie Thompson. 2022. A guide to abductive thematic analysis. *The Qualitative Report* 27, 5 (2022), 1410–1421. https://doi.org/10.46743/2160-3715/2022.5340

[8] Stefan Timmermans and Iddo Tavory. 2012. Theory Construction in Qualitative Research: From Grounded Theory to Abductive Analysis. *Sociological Theory* 30, 3 (Sept. 2012), 167–186. https://doi.org/10.1177/0735275112457914

Evaluating Large Language Model Code Generation as an Autograding Mechanism for "Explain in Plain English" Questions

David H. Smith IV
University of Illinois
Urbana, IL, USA
dhsmith2@illinois.edu

Craig Zilles
University of Illinois
Urbana, IL, USA
zilles@illinois.edu

ABSTRACT

The ability of students to "Explain in Plain English" (EiPE) the purpose of code is a critical skill for students in introductory programming courses to develop. EiPE questions serve as both a mechanism for students to develop and demonstrate code comprehension skills. However, evaluating this skill has been challenging as manual grading is time consuming and not easily automated. The process of constructing a prompt for the purposes of code generation for a Large Language Model, such OpenAI's GPT-4, bears a striking resemblance to constructing EiPE responses. In this paper, we explore the potential of using test cases run on code generated by GPT-4 from students' EiPE responses as a grading mechanism for EiPE questions. We applied this proposed grading method to a corpus of EiPE responses collected from past exams, then measured agreement between the results of this grading method and human graders. Overall, we find moderate agreement between the human raters and the results of the unit tests run on the generated code. This appears to be attributable to GPT-4's code generation being more lenient than human graders on low-level descriptions of code.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

GPT-4, Large Language Models, EiPE, Autograding

ACM Reference Format:
David H. Smith IV and Craig Zilles. 2024. Evaluating Large Language Model Code Generation as an Autograding Mechanism for "Explain in Plain English" Questions. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635542

1 INTRODUCTION AND BACKGROUND

The emergence and widespread popularity of services powered by "Large Language Models" (LLMs) has sparked concerns amongst educators. These concerns appear to primarily stem from the ease with which these models can be used to generate solutions to queries from a diverse set of topics [2]. Despite these concerns, there is equal excitement for the potential revolution in education this technology might bring about, particularly in the domain of computer science education [3]. With the rapid adoption of LLM powered tools like GitHub Copilot and ChatGPT, success as a programmer may increasingly hinge on one's ability to effectively employ these tools and evaluate the results they produce.

One such area of computer science education that may benefit from LLMs is the automatic grading of "Explain in Plain English" (EiPE) questions. In these questions, students are shown a segment of code and asked to demonstrate their comprehension of that code by providing a high-level description of its functionality [7]. EiPE questions typically require short answer responses their grading has often been manual which limits the ability for students to practice and receive timely feedback. Existing EiPE autograders take steps toward addressing this by giving students immediate feedback on the correctness of a response [4]. Though these graders perform similarly to a trained teaching assistant, to develop each EiPE question a large quantity of responses must be labeled which leads to overhead in the question authoring process.

To address this these limitations, we propose "Code Generation Based Grading" (CGBG). This grading approach uses a student's EiPE response to generate code via an LLM. The generated code is then tested for correctness through unit tests to determine if a student's description of a segment of code was able to generate functionally equivalant code. In this way, CGBG enables feedback by providing students the generated code and the results of the unit tests. It also simplifies the question authoring process by eliminating the need for human data labeling and replacing it with the simple authoring of unit tests. As an initial step towards evaluating CGBG's effectiveness as an EiPE grading mechanism we evaluate several implementations of CGBG via the following research questions:

RQ1 What is the agreement between trained human raters and code generation based grading?
RQ2 What relationships exist between the features of a given question and the agreement on that question?

2 CODE GENERATION BASED GRADING

The "Code Generation Based Grading" (CGBG) process we propose is divided into three distinct steps [8]. First, a student's response to an EiPE question and a pre-prompt instructing the LLM to generate a function based on the description are combined. This combined prompt is used to generate a function via GPT-4 which is then run against a set of manually defined unit tests. The student's grades for their response is then dependent on the success of the unit tests.

This process simplifies the question authoring process for producing automatically graded EiPE questions to be more similar to that of automatically graded code writing questions. This authoring

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635542

process is simpler than prior EiPE autograders, which require the creation of human labeled datasets and training of NLP models [1]. This also offers the benefit of allowing the instructor to concretely define what aspects of the code and edge cases should be described in the prompt through the unit tests.

However, a major consideration with this grading approach is how to account for the non-deterministic nature of GPT-4's responses. Through the model's temperature parameter, the "creativity" of the model can be controlled. A prompt made with a temperature of zero will always produce the same response. By increasing the temperature value the responses will gradually become less deterministic and more creative with the model's interpretation of the prompt. For the purposes of grading, we want to account for the possibility of a student being marked incorrect not because their prompt was poorly formed but simply because the model generated an overly creative response. With this in mind, we compare the following three grading approaches:

- **Single Response** A single response at 0.0 temperature is used for grading.
- **Best of 5:** Five responses are generated at 0.5 temperature. A prompt is graded as correct if at least one response passes all unit tests.
- **Majority Vote:** Five responses are generated at 0.5 temperature. A prompt is graded as correct if at least 3 of the 5 responses pass all unit tests.

3 METHODS

To evaluate the alignment between human graders and each of the proposed CGBG approaches, we conduct an analysis using historical data from a large introductory Python course taught at a large university in the United States. EiPE questions are an integral part of the course with students receiving explicit instruction on how to form adequate responses. When these questions appear on exams, they are graded manually by two trained teaching assistants (TAs). These TAs independently assessed responses using a rubric which covered which evaluated responses on three dimensions: 1) the functional correctness of the student's description, 2) a lack of ambiguity in the description, and 3) the description providing a high-level description of the codes purpose rather than a "line-by-line" description. After grading the responses the TAs met to consolidate their grades and reconcile any differences. A total of 6380 responses to 42 EiPE questions from past exams were subject to each of the three CGBG approaches. Cohen's $\varkappa$ was used to measure agreement between the human graders and each of the CGBG approaches across the 42 questions [6].

4 RESULTS AND DISCUSSION

The CGBG approaches introduced in this paper all achieve a moderate agreement with human raters with very little difference between them (Figure 1). Closely analyzing which questions fell into categories of high ($\kappa \geq 0.6$), medium ($0.4 < \kappa < 0.6$), and low ($\kappa < 0.4$) agreement revealed the primarily limitation of this approach is its inability to distinguish between high and low level descriptions. This limitation is most prominent on EiPE questions that require students to describe a short segment of code (1-2 lines). For example, consider the following segment of code:

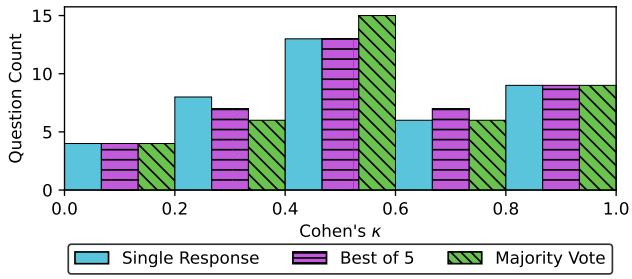


Figure 1: The distribution of agreements between each CGBG approach and human graders for each of the EiPE questions being evaluated.

```
def foo(x):
    return x % 2 == 0
```

A low level description might be "*determines if x mod 2 equals 0*" whereas a high level description would be "*determines if x is even*". As such, CGBG might be best suited for cases where instructors don't place as strong an emphasis on obtaining high-level descriptions. It may also be well suited for longer, more complex segments of code where heuristics such as limiting the length of students' responses could be used to nudge students towards higher level descriptions of the code.

There are several affordances this grading approach provides. Historically, deploying EiPE questions and evaluating their responses has either involved manual grading or the training of a model for each question. The use of CGBG reduces the authoring process to creating a sample code segment and a series of test cases. EiPE autograders have also faced challenges relating to a lack of transparency in their underlying grading mechanisms which can cause students to distrust the results they produce [5]. The proposed CGBG approaches aim to address this issue by providing students with the generated code as feedback along with the results of tests cases run on that same code. Future work will explore the impact of this feedback on students' ability to correct short coming in their responses and improve subsequent submissions.

REFERENCES

[1] Sushmita Azad. 2020. *Lessons learnt developing and deploying grading mechanisms for EiPE code-reading questions in CS1 classes.* Ph. D. Dissertation.
[2] Debby RE Cotton, Peter A Cotton, and J Reuben Shipway. 2023. Chatting and cheating: Ensuring academic integrity in the era of ChatGPT. *Innovations in Education and Teaching International* (2023), 1–12.
[3] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2022. Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language. *arXiv preprint arXiv:2210.15157* (2022).
[4] Max Fowler, Binglin Chen, Sushmita Azad, Matthew West, and Craig Zilles. 2021. Autograding" Explain in Plain English" questions using NLP. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education.* 1163–1169.
[5] Silas Hsu, Tiffany Wenting Li, Zhilin Zhang, Max Fowler, Craig Zilles, and Karrie Karahalios. 2021. Attitudes surrounding an imperfect AI autograder. In *Proceedings of the 2021 CHI conference on human factors in computing systems.* 1–15.
[6] Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica* 22, 3 (2012), 276–282.
[7] Laurie Murphy, Renée McCauley, and Sue Fitzgerald. 2012. 'Explain in plain English'questions: implications for teaching. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education.* 385–390.
[8] David H Smith IV and Craig Zilles. 2023. Code Generation Based Grading: Evaluating an Auto-grading Mechanism for" Explain-in-Plain-English" Questions. *arXiv preprint arXiv:2311.14903* (2023).

Programming Plagiarism Detection with Learner Data

Yifan Song*
University of Illinois
Urbana-Champaign, USA
yifan33@illinois.edu

Yuanxin Wang*
Carnegie Mellon University
Pittsburgh, USA
yuanxinw@alumni.cmu.edu

Marshall An
Carnegie Mellon University
Pittsburgh, USA
haokanga@andrew.cmu.edu

Christopher Bogart
Carnegie Mellon University
Pittsburgh, USA
cbogart@cs.cmu.edu

Majd Sakr
Carnegie Mellon University
Pittsburgh, USA
msakr@cs.cmu.edu

ABSTRACT

Courses with programming assignments have long faced the issue of academic integrity violations (AIV) where cheating could harm the outcome of student learning. Checking code similarity in students' final submissions is a common way to mitigate this issue. But this single analysis is insufficient as 1) students can refactor their code to evade the check, 2) mere code similarity may not be strong enough evidence to support an AIV case, particularly for simpler assignments that may have similar solutions, and 3) code similarity cannot reveal much about the actual circumstances and behaviors of plagiarism. Due to the lack of supporting data or tools, many educators either abandon solving these challenges or rely on manual approaches that are not feasible at scale. In this paper, we propose a workflow to solve the above challenges for large programming classes by providing supporting evidence of cheating with additional learner data: detailed submission timelines with scores and source code. Running this workflow in a large advanced programming course over several years has helped us identify many cheating cases effectively and efficiently.

ACM Reference Format:
Yifan Song, Yuanxin Wang, Marshall An, Christopher Bogart, and Majd Sakr. 2024. Programming Plagiarism Detection with Learner Data. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635520

1 INTRODUCTION

AIVs severely threaten the learning outcome of courses. Without effective plagiarism detection, a course may attract students who favor taking shortcuts, and discourage students who work independently. Programming assignments take days or weeks to complete, making centralized proctoring impractical for a large class [1]. MOSS (Measure Of Software Similarity), which uses a winnowing local fingerprinting algorithm to compare code similarity [3], is one of the most popular tools to detect plagiarism for programming. Educators upload students' final submissions to MOSS and receive a report identifying student pairs with high similarities.

However, relying only on students' final submissions is often insufficient to support a convincing AIV case. In intro-level courses, assignments are often provided with code templates, so code similarity checks may give a high positive rate. Conversely, such checks may be bypassed when students refactor their copied code in the final submission. Finally, code similarity does not reveal how plagiarism happens, and the instructor must draw from other sources to confirm the responsibilities and behaviors of each student.

In this work, we propose a workflow that augments code similarity checks with rich learner data to overcome the above challenges. Our workflow first supplements the MOSS system by concatenating all history submissions of a student into a single file to catch cheating cases happening before final submissions. While previous work has successfully used learner data for timeline analysis to detect cheating on *quizzes* [2], in this work, our pipeline is developed to gather and process learner data for auto-graded *programming assignments*. Abnormal behaviors and plagiarism patterns can then be identified and analyzed through a visualization dashboard.

The proposed workflow is used in a graduate-level project-based course with 100+ students that has been offered over 20 semesters. This course contains programming projects with multiple tasks that are available to students for 1-2 weeks. Students can submit their solutions to an auto-grading server as many times as they need until the deadline. The auto-grader provides immediate feedback, allowing students to revise their solutions. It is common for students to submit multiple times and improve their total project score over subsequent submissions. All submissions are captured, along with the learner data including the source code, the timestamp, and the grade of each submission.

2 WORKFLOW DESIGN

The first part of our workflow augments the regular MOSS by consolidating students' early-version submissions and the final submission into a single file before uploading to MOSS. MOSS compares not only the final submissions, but the whole submission history to detect plagiarism. Over the past 8 semesters, we have successfully identified 51 cheating pairs, among which 19 refactored their codes before final submission.

Alongside augmented code similarity evidence, learner data, i.e. detailed student activity logs, are often available to instructors but overlooked, since manually browsing and interpreting such massive logs can be time-consuming as class sizes grow. Therefore, we built

*Both authors contributed equally to the paper

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635520

Submission Time	Student	Total
2021-02-18 14:05:24	A	0
2021-02-18 14:17:53	A	20
2021-02-18 14:30:52	A	20
2021-02-18 14:42:29	A	25
2021-02-18 14:57:23	B	25
2021-02-19 19:12:24	A	33
2021-02-20 16:00:40	A	28
2021-02-20 16:20:04	A	48
2021-02-20 16:33:44	B	35
2021-02-20 16:52:55	B	40
2021-02-20 17:10:10	B	48
2021-02-20 18:59:46	A	73
2021-02-20 19:53:17	A	73
2021-02-20 19:57:08	A	73
2021-02-20 19:58:38	B	73
2021-02-20 20:21:05	A	82
2021-02-20 20:24:31	A	82
2021-02-20 20:28:09	A	85
2021-02-20 21:40:27	B	82
2021-02-20 21:45:07	B	85

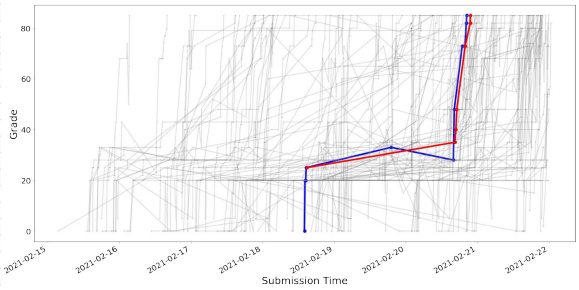


Figure 1: Left - activity log. Right - timeline plot for *Synchronous Submissions*. (blue:student A, red:student B, grey:rest of class)

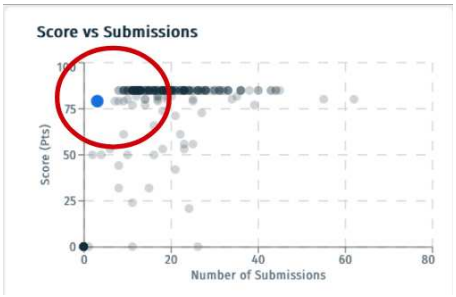


Figure 2: Number of submissions VS scores: the red circle in the zoomed-out plot shows suspicious cases with high scores but few submissions

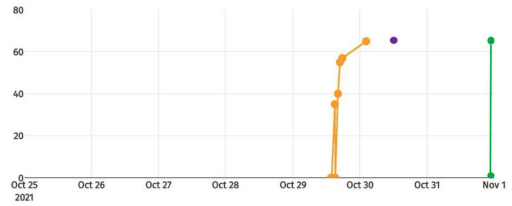


Figure 3: Pattern examples: *Normal Trajectory* (orange); *Too Few Submissions* (purple); *Late starter* (green)

a cloud-based pipeline to timely process the large-scale data to an interactive visual dashboard.

The dashboard then analyzes the learner data to detect anomalies which provides auxiliary evidence to support an AIV case. For example in Figure 2, visualizing the number of submissions and their scores across time in comparison to the entire class enables instructors to find outliers more efficiently and effectively.

In addition to salient individual outliers, more helpful insights can be found for pairwise activity data. In our projects, students can make multiple submissions and iteratively get feedback from an auto-grading service at different timestamps. Figure 1 shows the activity log table and the pairwise plot from an actual cheating case. The grey lines show how widely the student timelines varied for the rest of the class, providing stronger evidence for identifying unauthorized collaboration for a particular pair.

3 PLAGIARISM BEHAVIOR PATTERNS

Finally, we present some common types of cheating patterns, categorized through use of the dashboard in past semesters, that require attention from computer science educators.

Synchronous Submissions. The pair of students in Figure 1 achieved a similar grade for each task, with synchronous timestamps of submissions in the same time range, indicating that the two students could have worked together.

Too Few Submissions. For some projects that require intensive manual parameter tuning, it would take students many rounds of submissions to receive the feedback from auto-grader until a satisfactory score, even for experienced students. However, a student can copy parameters to achieve the full score with far fewer submissions. In Figure 3, the leftmost trace (orange) shows a typical student's trajectory with multiple attempts, while the middle (purple) trace achieved the full score with only one attempt.

Late Starters. This can be understood as a special version of *too few submissions*. While a project may normally require hours or days to complete, the rightmost (green) trace of Figure 3 shows a student who managed to start this assignment within the last hour before the deadline, and got a full mark for all tasks.

The ability to clearly describe such patterns has detected numerous cheating pairs in our courses, and has helped the course build a strong reputation of strict academic honesty.

REFERENCES

[1] Keith Adkins and David Joyner. 2022. Scaling anti-plagiarism efforts to meet the needs of large online computer science classes: Challenges, solutions, and recommendations. *Journal of Computer Assisted Learning* (07 2022). https://doi.org/10.1111/jcal.12710

[2] Jiameng Du, Yifan Song, Mingxiao An, Marshall An, Christopher Bogart, and Majd Sakr. 2022. Cheating Detection in Online Assessments via Timeline Analysis. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1* (Providence, RI, USA) *(SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 98–104. https://doi.org/10.1145/3478431.3499368

[3] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. 2003. Winnowing: Local Algorithms for Document Fingerprinting. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* (San Diego, California) *(SIGMOD '03)*. Association for Computing Machinery, New York, NY, USA, 76–85. https://doi.org/10.1145/872757.872770

See Logic In Life: An Individualized Exercise Structure for Making and Sharing Computing Concept Connections

Madalene Spezialetti
Computer Science Department
Trinity College
Hartford, Connecticut USA
madalene.spezialetti@trincoll.edu

ABSTRACT

Computing and computing concepts are interconnected with virtually every aspect of daily life and every field of study. However, when teaching new computing concepts to novice programmers, the number of example domain applications that can be presented is extremely limited. In addition, programming assignments where students apply these concepts are typically closed-ended, fully defining the problem to be solved. As a result, there are few opportunities for students to take an active role in creating connections between computing concepts and situations where they can be applied. *See Logic in Life* exercises are short, non-programming exercises comprised of a logical and a visual component. The exercise format provides opportunities for students to explore and create their own connections between computing concepts and application domains. Further, by sharing these connections with fellow classmates, students are exposed to a breadth of applications for a given concept via a peer learning experience. Student feedback on the impacts of both creating and sharing computing connections via *See Logic in Life* exercises is presented. Sample *See Logic in Life* exercises, suitable for use in CS0 or CS1 as in-class, out-of-class, or active learning exercises, are freely available at youtube.com/virt-university.

ACM Reference Format:
Madalene Spezialetti. 2024. See Logic In Life: An Individualized Exercise Structure for Making and Sharing Computing Concept Connections. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE '24). ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/3626253.3635616

1 FURTHER INFORMATION

When introducing novice programmers to new programming concepts, typical lecture examples and coding assignments fully define the application domain of the computing concepts they use and the output they produce. This closed-ended format eliminates the opportunity for students to contribute to framing the problem, choosing an application domain, or creating an end product that exhibits any individuality or differentiation. As a result, students are passive participants in defining connections between computing concepts and application domains.

Conversely, *See Logic in Life* (*SLiL*) exercises are designed to make students active participants in exploring, identifying, and creating collections of example applications that reflect computing concepts. These short, open-ended, non-programming exercises enable active student participation in:

Building Individual Connections with Computing Concepts. The SLiL exercises allow students to explore and create their own connections between computing concepts and the application of those concepts in "real-life" domains of their own choosing. This step opens the door to students choosing a domain of personal interest, the value of which has been demonstrated in [1] with media, in [4] with music, and in [2,5] with creativity. Further, the connection between giving students the opportunity to add unique contributions to open-ended specifications and increased confidence and satisfaction levels was shown in [3].

Using Individuality to Contribute to the Learning Experiences of Others. Sharing individual results allows students to make further computing connections by viewing the diversity of applications their classmates identified for the same construct. Sharing can take the form of class discussion, contribution to forums, or can be made via slideshows or videos. This step allows for a micro variation of peer instruction [6], in which everyone in the class contributes to and gains knowledge from the process.

1.1 See Logic in Life Exercise Structure

For each SLiL exercise, students are given a programming concept as a prompt and are instructed to connect it with a real-life example of their choosing that reflects the concept. The

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635616

assignment is structured to give each student two opportunities to creatively demonstrate their chosen application:

1) **Code-based component.** Students translate their chosen real-world example into a code or pseudocode snippet. This component provides an opportunity to make individual connections with the material and motivates students to review and digest new concepts.

(2) **Visual component.** Students submit an image representing the theme of their application. Choosing an image gives the student a second opportunity to show their individuality and facilitates readily sharing their ideas via a visual collection of sample applications, such as a video, slideshow or social media.

The SLiL structure is illustrated by the following example:

Identify a real-life example of an if-else statement and translate your example into a code/pseudocode Python snippet. Include an image that captures your example along with a title that summarizes its concept, as demonstrated by the code snippet and image of children on a roller coaster shown in Figure 1.

```
if (height >= rollercoaster_height_sign):
        print("Tall enough to ride coaster")
else:
        print("Not tall enough to ride coaster")
```



Figure 1: Illustration of Tall Enough to Ride the Roller Coaster.

Example if-else SLiL student submissions, which demonstrate the diversity of student application domains, include testing if a truck fits into a tunnel, checking if a banana is ripe, deciding to exercise a stock option, and testing if a credit score qualifies for a loan.

1.2 Impacts

A survey, approved by the IRB of Trinity College, explored the impact of the See Logic in Life exercises in a CS1 class consisting of 36 students. Seventeen of the students responded to the voluntary, anonymous year-end survey. In the class, the exercises were assigned as low-value homework and given throughout the semester. The exercises covered a variety of topics ranging from simple statements, such as if-else, to object-oriented topics.

A 5-option Likert scale (Strongly Disagree, Disagree, Neither Agree nor Disagree, Agree, Strongly Agree) was used to explore the impact of completing SLiL exercises and of viewing the submissions during class via slideshows. With regard to the benefits of creating their own SLiL examples, 88% of respondents either Agreed or Strongly Agreed that it helped them "*Better understand abstract computing examples by actively thinking of real-world examples*" and also to "*See examples of computing concepts in the real world*". When asked to comment on the benefits of developing their own examples, one student responded that it helped "*simplify concepts by rethinking them in more understandable contexts*". Yet another found the opportunity for individuality motivating, as they noted the exercises "*made me challenge myself since I always worked to come up with an idea not similar to the rest of my classmates*". The exercises allowed another "*to critically think about the logic…because that was necessary before trying to find a real-world application*".

With regard to the benefits of viewing slideshows of the SLiL submissions from the entire class, 83% of students Agreed or Strongly Agreed that it helped them "*Appreciate the breadth of ways the same computing concept could be applied*". When asked to describe the benefits, one student commented, "*I see other people's thoughts which helps me develop my own thinking.*" Another student responded, "*It helped me appreciate the creativity in the class by allowing me to see things from multiple perspectives*". Some students also noted the opportunity it provided to see common ground, such as the student who commented "*It was fun and interesting to see the variety of ideas…as well as how each See Logic in Life resulted in similar themes between responses*".

Ready-to-use See Logic in Life exercises, which include a prompt and an example code/pseudocode snippet with an image, are available for download at youtube.com/virt-university. The editable samples are suitable for the CS0- or CS1-level and can be readily adapted for different programming languages or for use as in-class, out-of-class, or active learning exercises.

REFERENCES

[1] Mark Guzdial. 2013. Exploring Hypothesis about Media Computation. ICER'13: Proceedings of the ninth annual international ACM conference on International Computing Education Research, August 2013, pages 19-26. https://doi.org/10.1145/2493394.2493397

[2] Markeya X. Pateranetz, Leen-Kiat Soh, and Elizabeth Ingraham. 2019. Building Computational Creativity in an Online Course for Non-Majors. SIGCSE '19: Proceedings of the 50th ACM Technical Symposium on Computer Science Education. February 2019, pages 442-448. https://doi.org/10.1145/3287324.3287346

[3] Sadia Sharmin, Daniel Zingaro, and Clare Brett. 2020. Weekly Open-Ended Exercises and Student Motivation in CS1. In Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research (Koli Calling '20), November 19-22, 2020, Koli Finland, ACM, New Your, NY, USA, 10 pages. https://doi.org/10.1145/3428029.3428035.

[4] Sebastien Siva, Tacksoo Im, Tom McKlin, Jason Freeman and Brian Magerko. 2018. Using Music to Engage Students in an Introductory Undergraduate Programming Course for Non-Majors. SIGCSE '18: Proceedings of the 49th ACM Technical Symposium on Computer Science Education. February 2018, pages 975-980. https://doi.org/10.1145/3159450.3159468

[5] Leen-Kiat Soh, Duane F. Shell, Elizabeth Ingraham, Stephen Ramsay and Brian Moore. 2015. Learning through computational creativity. Communications of the ACM, 8(Aug. 2015), 33-35. https://doi.org/10.1145/2699391

[6] Daniel Zingaro. 2014. Peer Instruction Contributes to Self-efficacy in CS1. SIGCSE '14: Proceedings of the 45th ACM Technical Symposium on Computer Science Education. March 2014, pages 373-378. https://doi.org/10.1145/2538862.2538878

Cracking the Cultural Code: Understanding the Cultural Barriers for Asian International CS Students in the US

Sandeep Sthapit
ssthapi@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Madison Thomas
mthoma27@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Janet Brock
jdbrock@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Tiffany Barnes
tmbarnes@ncsu.edu
North Carolina State University
Raleigh, NC, USA

ABSTRACT

In the field of computer science, cultural assumptions are embedded in programming languages and problem prompts. However, for students studying computer science in a Western country not from a Western culture, these assumptions can create learning barriers. This paper investigates the impact of cultural assumptions on international students studying computer science in a Western country; with a focus on understanding the barriers students face, how they overcome these barriers, and how barriers can be avoided. By performing thematic analysis on semi-structured interviews with 12 international graduate students at North Carolina State University, the authors found six main themes: Barriers, Increased Work, Emotions, Educational Environment, Overcoming Barriers, and Solutions to Barriers. Analyzing these themes provided insight into what barriers international students face and how they can be alleviated. The most common suggestions participants gave for alleviating barriers were more discussions around problem statements. By shedding light on this topic, the authors hope to inform computer science educators and researchers on the importance of creating inclusive and culturally relevant learning environments that accommodate the needs of diverse student populations.

CCS CONCEPTS

• **Social and professional topics → Computer science education**.

KEYWORDS

cultural identity, culturally relevant computing, international students, inclusive computing curricula and pedagogy

ACM Reference Format:
Sandeep Sthapit, Madison Thomas, Janet Brock, and Tiffany Barnes. 2024. Cracking the Cultural Code: Understanding the Cultural Barriers for Asian International CS Students in the US. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635567

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635567

Motivation. Instructors in the US often overlook inherent cultural assumptions. Such assumptions become barriers that impact the learning and experiences of international students studying Computer Science (CS). According to the Institute of Education Sciences, 900,000 college and university students are non US residents [6]; and according to the National Center for Science and Engineering Statistics, around 40% of graduate students (both master's and doctoral) are temporary visa holders [2]. Thus, there is a need to understand the barriers faced by international Students and how they overcome them to make CS more inclusive and culturally relevant.
Background. In order to understand the barriers faced by international students, previous research found that novice Non-Native English Speakers (NNES) international graduate students had greater confidence in reading versus writing or speaking [7]. Similarly, a study on novice programmers found that NNES face challenges with the instructional materials, technical communication, and reading and writing code [5]. Similarly, Soosairaj found that NNES students were stressed due to self-doubt and embarrassment from asking for help [3].
Research Methods We conducted 12 semi-structured interviews with graduate students in CS at North Carolina State University. Participants were recruited through the university's CS department; with a department wide email including a pre-screening survey. Eligible participants were invited for a recorded Zoom interview. The eligibility criteria included being 18 years or older, having some programming experience, being born and completing high school outside the United States, consenting to participate, and having faced a cultural barrier in computing. The audio transcripts of the interviews were analyzed using the six-step reflexive thematic analysis process described by Braun et al [4].
Participants We had 12 participants: 8 doctoral students, 4 master's students; 4 female and 8 male; 10 from South Asia and 2 from East Asia; all were at least bilingual; 11 completed undergraduate in their home country and 1 completed undergraduate in the US; 6 completed undergraduate in CS, while others were from related STEM majors; 6 had 4-8 years of programming experience and 7 participants rated themselves 4 out of 5 and rest rated themselves 3 out of 5 on programming proficiency. They also had high English proficiency, the average TOEFL score for graduate students at North

Carolina State University being 106 out of 120, with a minimum requirement of 18 in all four categories [1].

Findings

RQ1: What cultural barriers do international students face when studying computer science in the U.S.? The most commonly mentioned barriers were language barriers. Multiple participants mentioned struggling to understand accents or fast spoken English, which affected their ability to participate in class. P8 described how using their native language on their computer caused teaching staff to avoid them, which forced them to reach out to people who spoke their native language for help; P12 had similar experiences. This led them to change the language on their devices to English. When P12 was asked if they still get errors when typing their native language into an IDE, an issue they often faced in their undergraduate study, they said, "I don't even try and enter [my native language] in the IDE.." These communication issues affected participants ability collaborate on group projects. Many participants also mentioned feeling diffident when speaking English.

Another barrier participants faced was cultural differences. This frequently came in unfamiliar assignment contexts. P7 mentioned struggling with an Object Oriented Programming example centered around Western food. They said, "Ah, you will laugh because [the examples] were about pizza, toppings, and sides." This quote highlights how not knowing cultural context can be embarrassing for students, making them less likely to share their struggles and ask for help. Participants also discussed how work culture is different in the US. P1 and P3 noted that in the US, they had to schedule a meeting with their peers to seek help, but in their native country, they would reach out to their peers informally. This delayed their work and impacted their productivity. Interestingly, many participants expressed they did not realize they had faced cultural barriers when studying CS until they reflected on their experiences later.

Participants also faced barriers when it came to accessing certain resources. Multiple participants mentioned difficulties searching for solutions or coding advice due to a lack of resources in their native language. Participants also had trouble accessing certain technologies. P6 discussed how Arduinos were very expensive in their home country and how cutting-edge LLMs, like GPT4, often don't work well for languages besides English.

Differences in educational environments also caused barriers for many participants. Participants mentioned that CS curriculums in the US were different from their home countries. They described classes in the US as being more focused on applied work and having more assignments, whereas in their home countries their final grades mostly came from tests. Participants also noted dynamics with instructors were different in the US. Participants generally preferred instructors in the US compared to their home countries and found the instructors in the US more approachable, supportive, and knowledgeable in their respective courses while being careful not to single students out.

As a result of the aforementioned barriers our participants faced, they had an increased work load. Sometimes they would also face unexpected errors; P6 shared a story about creating a line following robot from an online tutorial. Upon debugging their robot's code, they found that the tutorial's hidden assumption of imperial system measurements caused the bug. The extra work and barriers caused many of our participants to feel frustrated. They expressed negative emotions to us, P2 said they felt "terrified" because they believed they did not know as much as the other students in their class.

RQ2: How do students overcome these barriers? "Yeah, of course, because that was what I had to do.", P12 expresses their persistence and resilience when asked if they overcame the barriers they faced. Being graduate students, all of our participants displayed similar resilience to overcome these barriers. The three main ways our participants overcame barriers were: 1) Helping themselves: Participants read more or looked for the solutions themselves, such as P5 who watched StarWars to understand cultural references to R2D2 in an online code repository. 2) Online Help: Participants used online forums, documentation, videos from YouTube, or communities like Reddit. For example, P12 received help from an online developer community that spoke their language when faced with errors caused by using their native language in her computer program. 3) Help from others: Participants received help from their peers, instructors, or advisors. Peer help was related to learning programming concepts as well as sharing struggles and frustrations.

RQ3:What could be done to alleviate these barriers? Most of our participants' suggestions involved the role of the instructor. Nearly half of the participants suggested having clear classroom discussions around programming problems would help with unfamiliar contexts. Instructors could explain cultural context as suggested by P7, or they could use more universal cultural contexts relevant to student life, such as groceries, as suggested by P9; while P4 suggested translating or removing contexts altogether. P6 suggested explicitly stating and explaining cultural context on programming assignments and tutorials.

Limitation and Future Work. The main limitation of our work is the small sample size. All our participants were from Asian countries so we lacked perspectives from other cultures. Additionally all our participants were graduate students studying in the US so they all spoke English proficiently and had extensive CS experience. Future work should be done with a larger sample size, include participants from other cultures, include multiple institutions, and both students proficient and not proficient in English to compare experiences. The effectiveness of the suggestions participants made to alleviate the barriers should also be tested in future works. As CS and the number of international students in the US grows this work will become evermore important and more research should be done on how to lessen the cultural barriers students are currently facing.

REFERENCES

[1] [n. d.]. Graduate Program - GRE / English Test Requirements. https://www.csc.ncsu.edu/academics/graduate/gretoefl.php. Retrieved Nov 20, 2023.

[2] 2022. International students in the US (72 stats for 2023). https://opendoorsdata.org/annual-release/international-students/. Retrieved May 1, 2023.

[3] Vardhan Agarwal, Yada Chuengsatiansup, Elise Kim, Yuzi LYu, and Adalbert Gerald Soosai Raj. 2022. An Analysis of Stress and Sense of Belonging Among Native and Non-native English Speakers Learning Computer Science. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*. ACM, 376–382.

[4] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101.

[5] Philip J. Guo. 2018. Non-Native English Speakers Learning Computer Programming: Barriers, Desires, and Design Opportunities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 1–14.

[6] Alvin Parker. 2023. *International students in the US (72 stats for 2023).* Retrieved May 1, 2023.

[7] C Tardy. 2004. The role of English in scientific communication: lingua franca or Tyrannosaurus rex? *Journal of English for Academic Purposes* 3, 3 (2004), 247–269. https://doi.org/10.1016/j.jeap.2003.10.001

ThemeRec : Personalizing IDE Themes for Students

Jialiang Tan*
Lehigh University
Bethlehem, Pennsylvania, USA
jit623@lehigh.edu

Yu Chen
William & Mary
Williamsburg, Virginia, USA
ychen39@wm.edu

Shuyin Jiao
North Carolina State University
Raleigh, North Carolina, USA
sjiao2@ncsu.edu

ABSTRACT

Prolonged screen time while working on programming assignments can often lead to student fatigue, boredom, and stress. In response to these issues and to enhance the overall programming environment for students, we introduce ThemeRec , a Visual Studio Code extension. ThemeRec provides students with personalized recommendations for code themes, with a particular focus on semantic coloring combinations. It boasts an extensive collection of over 150 code themes, each offering unique combinations of background colors, font styles, and semantic color arrangements. Using an integrated collaborative filtering recommendation system, ThemeRec automatically suggests and installs various themes based on the preferences of similar users whenever they want to change a theme. To assess the effectiveness of ThemeRec , we conducted an evaluation within an introductory-level programming course. The feedback from students has been positive and encouraging. This underscores the potential value that ThemeRec brings to programming courses.

ACM Reference Format:
Jialiang Tan, Yu Chen, and Shuyin Jiao. 2024. ThemeRec : Personalizing IDE Themes for Students. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635577

1 INTRODUCTION AND RELATED WORK

In recent years, programming has become increasingly recognized as an essential skill for students across various fields. Nevertheless, mastering programming is not an easy feat; it requires dedicated learning and practice. The arduous process of spending hours on programming assignments can often leave students feeling bored, fatigued, and/or stressed. Consequently, many students lose interest in programming, resulting in a high dropout rate [1].

With the development of e-learning, researchers and educators have worked diligently to enhance students' digital learning environments [2]. In computer science education, integrated development environments (IDEs) have played a pivotal role in improving students' coding experiences. IDEs like Visual Studio Code (VS Code)[1] and Spyder[2] offer features such as syntax understanding, build automation, code linting, and debugging, significantly enhancing the efficiency and accessibility of programming. As a result, instructors increasingly incorporate IDEs into introductory programming courses. However, IDEs typically come with default color themes that may not cater to every student's preference. For instance, VS Code uses a default black background color, but some students prefer a colorful background. Customizing the color settings is not always straightforward, especially for entry-level students.

In this study, we developed an open-source tool named ThemeRec , with the primary goal of improving students' visual comfort and increasing interests in programming. The main focus of this study was to address the following research questions:

RQ 1. Does ThemeRec effectively assist students in reducing levels of visual fatigue, anxiety, and stress while engaging in programming activities?
RQ 2. Does ThemeRec contribute to improving students' focus and programming productivity?
RQ 3. Are there discernible differences in the perception and utilization of ThemeRec among students with varying levels of coding experience and different genders?

2 THEMEREC 'S DESIGN

ThemeRec is developed as a VS Code extension. Its primary function is to provide students with personalized recommendations for semantic coloring code themes within VS Code. The overarching goal is to mitigate extended screen fatigue and bolster engagement during coding activities. To achieve this, we've curated an extensive collection of more than 150 semantic coloring code themes sourced from the marketplace[3]. Each theme in this collection offers unique combinations of background colors, font styles, and code element colors based on their semantic meaning and role within the code.

As illustrated in Figure 1, upon installation users immediately spot two extra buttons integrated into VS Code. The *refresh* button is dedicated to suggesting diverse code themes from our theme pool, leveraging our intelligent recommendation algorithms. ThemeRec adopts two recommendation algorithms (described as two modes below). With each click, it automatically applies a recommended theme within VS Code. Conversely, the *default* button serves as a rapid means to revert to the default theme settings of VS Code.

ThemeRec employs two recommendation modes: easy and complex. In easy mode, themes are ranked by popularity based on averaged installs. In complex mode, ThemeRec adopts the collaborative filtering recommendation algorithm [3], which recommends themes to user A based on the interests of a similar user B. For every theme in the theme pool, we extracted eight features: background color, dark/light tone, keyword color, font, high/low contrast, publisher, installs, m_rating (averaged rating on the marketplace). To

*This work is done when Jialiang visits at North Carolina State University.
[1]https://code.visualstudio.com/
[2]https://www.spyder-ide.org/
[3]https://marketplace.visualstudio.com/

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635577

Table 1: Questions and collected responses in the ThemeRec 's experience survey.

Questions	Yes	No	N/A
Q1. Do you like the default VS Code theme?	2	4	2
Q2. Do you have trouble installing ThemeRec ?	1*	6	1
Q3. Are you satisfied with ThemeRec 's installation?	7	0	1
Q4. Are you satisfied with the way ThemeRec works?	8	0	0
Q5. Do you like the themes recommended by ThemeRec ?	8	0	0
Q6. How often do you refresh your themes via ThemeRec ?**	-	-	-
Q7. Typically, when do you refresh your theme?**	-	-	-
Q8. Did you find your favorite theme from ThemeRec 's recommendations?	5	1	2
Q9. Do you think changing themes (via ThemeRec) adds to your interest in programming?	4	3	1
Q10. Do you think changing themes (via ThemeRec) increases your productivity?	2	2	4

*: We resolved the installation issue for this student by upgrading their outdated version of VS Code.
**: The statistics of Q6 and Q7 are described in Section 3.

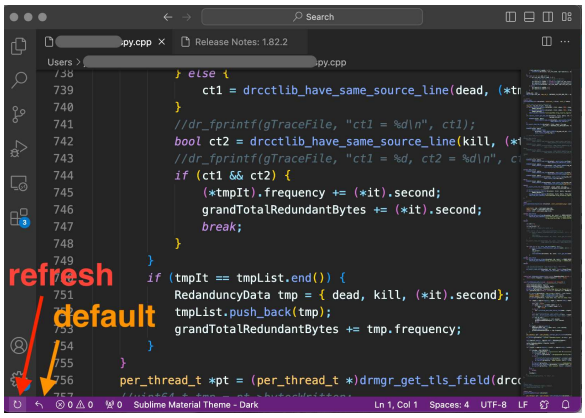


Figure 1: VS Code user interface upon ThemeRec 's installation. Refresh button and default button will be integrated into the status bar (the bottom purple bar), by clicking refresh/default button, ThemeRec automatically recommends and installs a new theme/changes back to the default theme. The corresponding theme name (*Sublime Material Theme - Dark*) appear on the status bar for user's reference and disappear after three minutes.

build the dataset, we invited students from various background, collected students' personal information, then we asked them to experience themes and rate themes from above eight perspectives. Based on the collected dataset, ThemeRec filters out themes that a user might like on the basis of reactions by similar users.

3 RESULTS

In Spring 2023, we conducted a study with students in an introductory Python course at North Carolina State University, where students use VS Code for coding assignments. Tutorial sessions were held later in the semester to guide ThemeRec installation and usage. At the end of the semester, we administered a survey aimed at gathering insights on their experiences and feedback regarding ThemeRec 's usage (because of limited participants, they only experienced ThemeRec 's easy mode). Due to the invitation sent late during the semester, only eight students participated the survey, who are freshmen and sophomores majoring in Science and Engineering. There was one male, six females, and one preferred not to disclose their gender. Despite the limited sample size, the feedback was highly positive.

The overall rating of ThemeRec in average is 4.25 (out of 5), the rest questions and responses are described in Table 1. Half of the students disliked the default themes (Q1), but they all like themes recommended by ThemeRec (Q5), and 62.5% found their favorite themes through ThemeRec 's recommendations (Q8). Their satisfactions of ThemeRec 's installation and functionality were significantly high (Q2-Q4), with an average installation time of under 5 minutes. For the statistics of Q6, 62.5% of the students periodically refreshed their themes every 2-3 days, 25% refreshed 2-5 themes per day, while 12.5% students refreshed themes frequently, more than 10 themes per day. For the statistics of Q7, 25% students preferred to refresh theme at the start of coding sessions, while 75% students refreshed their themes when feeling fatigued or bored during programming sessions. However, the subjective assessment of the impact of theme changes on student interest and productivity was inconclusive (Q9, Q10), requiring further objective evaluation.

4 FUTURE WORK

As an ongoing endeavor, we are committed to further enhancing ThemeRec . We plan to introduce new functionalities that enable effective monitoring of students' usages. These features will record various aspects of user interaction, including the timing of clicks on the theme-switching button, the specific theme in use, the file type they are working on, and their active engagement time with the file. These data points will form the basis for developing a more intelligent algorithm capable of tailoring theme recommendation to individual preferences and needs. Our goal is to create a more personalized and engaging coding environment for every user.

REFERENCES

[1] Nigel Bosch and Sidney D'Mello. 2017. The affective experience of novice computer programmers. *International journal of artificial intelligence in education* 27 (2017), 181–206.
[2] Rita Garcia, Katrina Falkner, and Rebecca Vivian. 2018. Systematic literature review: Self-Regulated Learning strategies using e-learning tools for Computer Science. *Computers & Education* 123 (2018), 150–163.
[3] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web: methods and strategies of web personalization*. Springer, 291–324.

Supporting Mastery Learning with Flexible Extensions

Yuerou Tang*
University of California Berkeley
Berkeley, CA, USA
yuerou.tang@berkeley.edu

Jacob Yim*
University of California Berkeley
Berkeley, CA, USA
jacobyim@berkeley.edu

Jordan Schwartz
University of California Berkeley
Berkeley, CA, USA
jordanschwartz@berkeley.edu

Madison Bohannan
University of California Berkeley
Berkeley, CA, USA
mkcb@berkeley.edu

Dana Benedicto
University of California Berkeley
Berkeley, CA, USA
dbenedicto@berkeley.edu

Charisse Liu
University of California Berkeley
Berkeley, CA, USA
charisseliu@berkeley.edu

Armando Fox
University of California Berkeley
Berkeley, CA, USA
fox@berkeley.edu

Lisa Yan
University of California Berkeley
Berkeley, CA, USA
yanlisa@berkeley.edu

Narges Norouzi
University of California Berkeley
Berkeley, CA, USA
norouzi@berkeley.edu

ABSTRACT

Equitable grading practices and flexible deadline policies have previously demonstrated positive student learning and well-being outcomes. In this poster, we contribute a framework for flexible extension policies that emphasize equitable grading. We then analyze extension requests and grades obtained by students in a data science course with a flexible extension policy. We present two research questions based on this data. **RQ1:** How does the length of an extension relate to student performance on the corresponding assignment? **RQ2:** How does student extension usage across the semester relate to students' learning of the content?

KEYWORDS

Equitable grading, Flexible extension policy, Mastery learning

ACM Reference Format:
Yuerou Tang, Jacob Yim, Jordan Schwartz, Madison Bohannan, Dana Benedicto, Charisse Liu, Armando Fox, Lisa Yan, and Narges Norouzi. 2024. Supporting Mastery Learning with Flexible Extensions. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635615

1 INTRODUCTION

In a traditional classroom setting, students are subject to rigid due dates. Credit is given to students for work that is submitted in a timely manner. This system, however, tends to favor those with prior experience–students with experience tend to be more prepared and need less time to complete the assignments, leading to concerns about the equity and effectiveness of the system [1, 3].

An alternative approach, *mastery learning*, allows students to attain any grade they are willing to put in the time and effort for [3]. This often involves giving some students more time on specific assignments and incorporating an auto-grader so that students can get instant feedback on their work. Such practices allow students to master concepts more thoroughly, even if it means that they would need more time. Prior research also shows that such mastery learning practices have a positive effect on student engagement, learning progress, and final grades [2, 4]. This paper examines a data science course incorporating a mastery learning approach into its grading policies.

2 CLASSROOM DESCRIPTION

The study is conducted in a data science course intended for students with previous beginner-level programming and statistics experience, with a focus on fundamental techniques of data science. The study was performed during Spring 2023 with an enrollment of 1225 students.

Take-home assignments consisted of homework assignments, projects, and labs. Labs were short, low-stakes assignments, while homework assignments and projects were longer assignments worth a more significant percentage of students' grades. There is a pre-configured auto-grader written by the course staff for assignments involving programming. The submission platform also outputs the score immediately after submitting the assignment. For assignments with written questions, course staff manually graded the written portion. All auto-grader test cases for labs were visible to students, while homework and projects incorporated hidden tests.

3 POLICIES

To make mastery learning a reality, we propose a set of policies centered around flexible extensions on assignments. These policies were implemented in the data science course described in Section 2. Figure 1 provides an overview of student experience with flexible extensions in our classroom model.

To address unexpected student emergencies during the course, instructors provided an Extenuating Circumstances Google Form.

*The first two authors contributed equally to this research.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635615

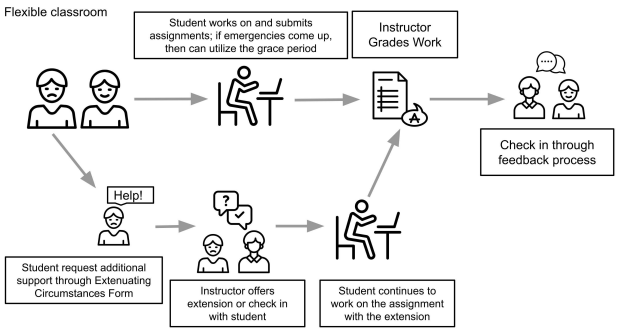


Figure 1: Student experience with flexible assignment extensions.

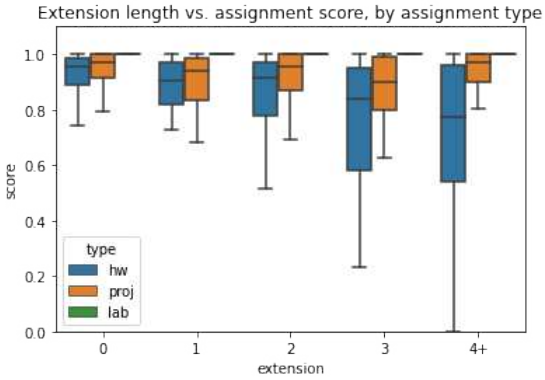


Figure 2: Association of assignment scores by extension length for different assignment types.

Students were encouraged to fill out this form to account for extenuating circumstances, such as personal health issues or family emergencies, preventing them from completing the assignment. The form asked for the assignment on which the student would like to request an extension, the reasons for the extension request, the proposed extension duration, and the student's availability in case a meeting was necessary. Students were given two days for each request unless the student explicitly requested more.

For each student's form submission, course staff approved or denied the extension request based on the reason outlined by the student and their extension request history. Email updates were then sent to students based on the decision. If a student repeatedly requested extensions on assignments, course staff reached out to schedule a 30-minute meeting to check in with the student on their learning progress, answer any questions, and provide advice.

4 RESULTS AND DISCUSSION

To address RQ1, we first examine the correlation between extension lengths and student grades on the extended assignments. Figure 2 displays the association between extension lengths and grades on corresponding assignments. For homework assignments, as extension lengths increase, the median student score on corresponding submissions tends to decrease. This trend could result from students requesting extensions on assignments they already struggled with. On the other hand, for labs, extension lengths appear to have little to no effect on student scores; most students receive near-perfect scores on labs regardless of extension length, possibly due to the visible test cases and lower difficulty of lab assignments.

To address RQ2, we evaluate the correlation between extensions across the duration of the semester and student final exam scores, using final exam performance as an indicator of student understanding of the course content. We discovered a negative correlation of -0.9 between these two variables; students requesting more extension days tend to receive lower grades on the final exam. Similarly to extensions on individual assignments, heavier use of extensions throughout the semester may indicate that students struggle with the material or with recurring extenuating circumstances impacting their learning of the content.

5 FUTURE WORK

Moving forward, we plan to build on our analysis to more clearly assess the effects of a flexible extensions policy. Our current data about student grades provides limited information about student learning gains, student well-being, and the trade-offs associated with alternative policies. Rather than just grades, we plan to analyze the results of surveys administered to students at the end of the semester. We also hope to use focus group interviews to better understand the effects on student learning and well-being, especially for students with disability accommodations and students from underrepresented groups. We would like to conduct similar analyses on different classes to compare the effects of different policies, as undergraduate data science and computer science classes incorporate various policies surrounding extensions. In particular, we would like to compare data from classes with flexible extensions to classes without extensions. This may allow us to determine the effects of an extension policy on exam scores, grade distributions, survey results, the number of students dropping the class, and other metrics of student learning gains and well-being.

We are also currently developing a tool to automate the process of granting extensions for instructors, reducing the effort involved with implementing a flexible extension policy. This tool is discussed in greater detail in a different work [5].

REFERENCES

[1] Andrew Berns, J Philip East, and J Ben Schafer. 2021. Grading for Equity: A Curriculum Development and Grading Process to Enhance Instruction. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 1351–1351.

[2] Huanyi Chen and Paul AS Ward. 2023. The Value of Time Extensions in Identifying Students Abilities. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. 512–518.

[3] Dan Garcia, Connor McMahon, Yuan Garcia, Matthew West, and Craig Zilles. 2022. Achieving" A's for All (as Time and Interest Allow)". In *Proceedings of the Ninth ACM Conference on Learning@ Scale*. 255–258.

[4] Claudia Ott, Brendan McCane, and Nick Meek. 2018. Five years of Mastery Learning: What did we learn?. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*. 1–2.

[5] Jordan Schwartz, Madison Bohannan, Jacob Yim, Yuerou Tang, Dana Benedicto, Charisse Liu, Armando Fox, Lisa Yan, and Narges Norouzi. 2024. Automated Support for Flexible Extensions. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education*.

Learning Software Engineering Principles with Program Wars *v.*3.0

Md. Hasan Tareque
mdhasan.tareque@uleth.ca
University of Lethbridge
Lethbridge, Alberta, Canada

John Anvik*
john.anvik@uleth.ca
University of Lethbridge
Lethbridge, Alberta, Canada

ABSTRACT

Program Wars is a web-based card game for teaching fundamental concepts of programming and cybersecurity. We propose Program Wars *v.*3.0, an extension to Program Wars *v.*2.0, to explore the effectiveness of teaching fundamental software engineering concepts using GBL.

In Program Wars *v.*3.0, a player constructs card decks to complete three 'sprints' based on the specific requirements for the sprint. After playing a fixed number of turns, in which the player builds a pseudo-program, the player's program is tested for requirement completion and either a new sprint begins with additional requirements or the game ends.

Program Wars *v.*3.0 differs from prior work by allowing the user to learn and practice software engineering principles in an engaging manner and within a shorter time frame than traditional methods.

ACM Reference Format:
Md. Hasan Tareque and John Anvik. 2024. Learning Software Engineering Principles with Program Wars *v.*3.0 . In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635551

1 MOTIVATION

Experience-based education plays a vital role in the computer science (CS) curriculum, as learning through activities that combine social and educational aspects is shown to be very effective [9]. However, constraints such as time, proper tools and realistic environments introduce difficulties to such an approach. Game-Based Learning (GBL) has been used as an effective way of teaching concepts practically, particularly the use of serious games [8]. Serious games focus on the usefulness of the entire gameplay in the context of learning, and its introduction into a CS curriculum is known to be an effective one [4], having been used to teach such areas as debugging [10] and software project management [6].

Program Wars [1, 3] is a web-based card game for teaching fundamental concepts of programming (i.e. variables, loops, methods) and cybersecurity (i.e. malware, cyberattacks, firewalls, antivirus). A full description of Program Wars *v.*2.0 is found in [13] and the game can be played at https://program-wars.firebaseapp.com

The game was created to address several problems with traditional approaches to teaching programming. These problems include: (1) requiring the student to either learn an existing programming language or a language specific to the learning environment, (2) educational programming languages and environments which are "often ambitious in what they are trying to teach beginning programmers" [5], (3) puzzle-solving activities that tend to favour a specific demographic [11], and (4) sandbox paradigms requiring oversight by an outside influence (i.e. an instructor) to ensure learning progression [5].

Program Wars *v.*3.0 is a planned extension for Program Wars *v.*2.0 to explore further the effectiveness of teaching software engineering concepts using GBL. Specifically, Program Wars *v.*2.0 is augmented to introduce the fundamental concepts of the Software Development Life Cycle (SDLC) and the Iterative Software Development Methodology (ISDM) (a.k.a. Agile software development). Although other card-based games have concentrated on teaching ISDM, these efforts focus either on a specific Agile methodology, such as Scrum, teamwork aspects, or both [2, 7, 12]. Program Wars *v.*3.0 differs in exposing the player to various software engineering concepts and allowing them to practice the concepts by implementing a representation of a computer program, thereby making it independent of any specific agile methodology or programming language.

2 PROGRAM WARS *V.*3.0

The primary goal of Program Wars *v.*3.0 is to provide experiential learning of the SDLC and ISDM concepts. In Program Wars *v.*3.0, the gameplay of Program Wars *v.*2.0 is changed to include a new mode, called `ISDM` mode. In the new mode, players still compete for points as in Program Wars *v.*2.0 , but the game is broken into three rounds called *sprints*. Each sprint is comprised of the four *SDLC* phases: *Requirements*, *Design*, *Implementation* and *Testing*. The game ends when all three sprints have been completed.

2.1 Requirements Phase

In this phase, players are given requirements based on the sprint goal. These requirements are sets of individual objectives that award points and other bonuses to the player when completed. The requirements represent the requirements that would be given to a developer or team by a customer for a new software project or as a part of an existing software project. Requirements carry over from one sprint to the next.

Requirement completion is checked in the *Testing* phase. If a player completes an objective by the end of the sprint it is associated with, the player receives an additional bonus. These bonuses

*We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number RGPIN-2018-06004].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635551

may include points but can also give the player a useful card or status effect. Some sprint bonuses are awarded immediately upon completing the objective, and others are awarded at the end of the sprint based on how they might help the player. Each sprint has its own requirements, as given below.

The player must complete at least two sprint requirements to win the game. This allows beginner players an opportunity to get used to the format of the game without excelling immediately. Players receive different rewards depending on using specific game cards during a sprint. The sprint bonuses make it easier for the player to get more points or to complete subsequent objectives before the end of the game.

Sprint #1 The player is to use the cards for methods (`Method`) and loops (`Repeat`) several times. This teaches the SE design principle of 'Do Not Repeat Yourself' (a.k.a. "D.R.Y.").
Sprint bonuses allow the player to play with an extra card during this sprint, gain 10 bonus points, and receive a cyberdefense card (`Computer Scan`) for use in the next sprint.

Sprint #2 The player is to play cyberdefense cards to secure the system from various hacking and malware attacks. This teaches the design principle of system security.
Sprint bonuses include 10, 20 or 30 bonus points depending on how many cyberdefense cards are played, gaining a card that increases the power of some `Repeat` cards, and the ability to discard and redraw their hand.

Sprint #3 The player is to play different cyberattack cards so that they can find and exploit vulnerabilities in an opponent's program. This teaches the testing practice of penetration testing.
Sprint bonuses include 10, 20 or 30 bonus points depending on how many successful cyberattacks they made on opponents, a card that allows the player to sort the top of the draw deck, and the ability to discard and redraw their hand.

2.2 Design Phase

After a player receives the requirements for a sprint, the player develops a customized deck based on the sprint requirements. As each sprint has different objectives, players must make choices about how they can best complete these objectives. This phase represents the part of a software project where developers make decisions about how the software will be built.

Players start with the base Program Wars cards. Then they choose a limited number of additional cards from a card pool to meet the sprint requirements. The card pool is unique for each sprint so that players can access cards that will be most useful for the requirements. Also, there are cards that may not be essential, but a player can include those cards as a part of their strategy. More powerful cards are limited in number and do not appear in the card pools for all sprints.

2.3 Implementation Phase

The *Implementation* phase, for the most part, is played the same as Program Wars *v.*2.0. The major change is that the game no longer ends when a player reaches a specific number of points. Instead, players have 10 turns to complete the sprint objective(s).

2.4 Testing Phase

The *Testing* phase represents the sprint acceptance testing for the built program, and the player is judged on their progress towards fulfilling sprint requirements. This is where bonus points and rewards for completing requirements are given.

After the *Testing* phase, the game state is saved, and players are returned to the *Requirement* phase for Sprint 2 or 3. After Sprint 3, the game ends, and the player who completed at least two sprint requirements with the most points is declared the winner.

3 CONTRIBUTIONS

Knowledge of the SDLC and ISDM is vital for those who seek to become software developers. With Program Wars *v.*3.0 , the concepts of the SDLC and ISDM are presented by integrating them into the implementation of Program Wars *v.*2.0. By introducing several sprints, Program Wars *v.*3.0 allows the player to experience the SDLC and ISDM in an engaging manner. Also, students gain this experience in a shorter time than traditional teaching methods, which may require weeks or months. The sprint themes also introduce system design principles, and security implementation and testing.

REFERENCES

[1] [n. d.]. Program Wars. https://program-wars.firebaseapp.com/. [Accessed Oct 13, 2023].

[2] [n. d.]. Scrum Card Game. https://www.tastycupcakes.org/2016/06/scrum-card-game/. [Online; accessed Oct, 13, 2023].

[3] John Anvik, Vincent Cote, and Jace Riehl. 2019. Program Wars: A Card Game for Learning Programming and Cybersecurity Concepts. In *50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. 393–399. https://doi.org/10.1145/3287324.3287496

[4] Alex Baker, Emily Oh Navarro, and André van der Hoek. 2005. An experimental card game for teaching software engineering processes. *J. Syst. Software* 75, 1 (2005), 3–16. https://doi.org/10.1016/j.jss.2004.02.033

[5] Kohl Bromwich, Masood Masoodian, and Bill Rogers. 2012. Crossing the Game Threshold: A System for Teaching Basic Programming Constructs. In *13th Int. Conf. NZ Chapter ACM's SIG on HCI* (Dunedin, New Zealand) *(CHINZ '12)*. 56–63. https://doi.org/10.1145/2379256.2379266

[6] Alejandro Calderón, Mercedes Ruiz, and Elena Orta. 2017. Integrating Serious Games as Learning Resources in a Software Project Management Course: The Case of ProDec. In *2017 IEEE/ACM 1st Int. Workshop on SE Curricula for Millennials (SECM)*. 21–27. https://doi.org/10.1109/SECM.2017.3

[7] João M. Fernandes and Sónia M. Sousa. 2010. PlayScrum - A Card Game to Learn the Scrum Agile Method. In *2nd Int. Conf. on Games and Virtual Worlds for Serious Applications*. 52–59. https://doi.org/10.1109/VS-GAMES.2010.24

[8] G. Bente J. S. Breuer. 2010. Why so serious? On the Relation of Serious Games and Learning. *Eludamos, Journal for Computer Game Culture* 4, 1 (2010), 7–24.

[9] Jacqueline E. Maloney, Molly Stewart Lawlor, Kimberly A. Schonert-Reichl, and Jenna Whitehead. 2016. *A Mindfulness-Based Social and Emotional Learning Curriculum for School-Aged Children: The MindUP Program.* Springer New York, 313–334. https://doi.org/10.1007/978-1-4939-3506-2_20

[10] Michael A. Miljanovic and Jeremy S. Bradbury. 2017. RoboBUG: A Serious Game for Learning Debugging Techniques. In *2017 ACM Conf. on Int. Computing Education Research (ICER '17)*. 93–100. https://doi.org/10.1145/3105726.3106173

[11] Mikki H. Phan, Jo R. Jardina, Sloane Hoyle, and Barbara S. Chaparro. 2012. Examining the Role of Gender in Video Game Usage, Preference, and Behavior. *Proc. of the Human Factors and Ergonomics Society Annual Meeting* 56, 1 (2012), 1496–1500. https://doi.org/10.1177/1071181312561297

[12] Guillermo Rodriguez, Álvaro Soria, and Marcelo Campo. 2015. Virtual Scrum: A teaching aid to introduce undergraduate software engineering students to scrum. *Comput. Appl. Eng. Educ.* 23, 1 (2015), 147–156. https://doi.org/10.1002/cae.21588

[13] Md. Hasan Tareque. 2021. *Updating a Web-based Card Game to Teach Programming, Cybersecurity and Software Development Life Cycle Concepts.* Master's thesis. University of Lethbridge.

Enhancing Programming Education through Game-Based Learning: Design and Implementation of a Puyo Puyo–Inspired Teaching Tool

Ruochen Tian
Waseda University
Tokyo, Japan
tianruochenjimmy@moegi.waseda.jp

Daisuke Saito
Takachiho University
Tokyo, Japan
d.saito@takachiho.ac.jp

Hironori Washizaki
Waseda University
Tokyo, Japan
washizaki@waseda.jp

Yoshiaki Fukazawa
Waseda University
Tokyo, Japan
fukazawa@waseda.jp

Hiroshi Koboyashi
System Information Co.,Ltd
Tokyo, Japan
kobayashi-hiroshi@sysj.co.jp

Ayumi Tsuji
System Information Co.,Ltd
Tokyo, Japan
tsuji-ayumi@sysj.co.jp

ABSTRACT

Although programming is part of primary school curricula in many countries, barriers persist for elementary students learning programming such as an insufficient understanding of the underlying mathematics, complex concepts, and purpose of programming. These challenges often lead to disinterest. Herein we present an innovative game-design–based programming education tool. Students progressively enhance a classic game, Puyo Puyo, using fundamental programming concepts and selecting the appropriate code. This engaging approach improves students' computational thinking abilities as they transform code into a functional game. Here, we describe the tool's background and structure. Then we detail a workshop using the tool, including analyzing the changes in students' programming skills, computational thinking, and interest in programming. Finally, we summarize the findings and future research directions.

CCS CONCEPTS

• **Social and professional topics** → **K-12 education**; **Computational thinking**.

KEYWORDS

K-12 education, block-based coding, game elements, computational thinking

ACM Reference Format:
Ruochen Tian, Daisuke Saito, Hironori Washizaki, Yoshiaki Fukazawa, Hiroshi Koboyashi, and Ayumi Tsuji. 2024. Enhancing Programming Education through Game-Based Learning: Design and Implementation of a Puyo Puyo–Inspired Teaching Tool. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635477

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635477

1 INTRODUCTION

In many countries, elementary curricula incorporate programming. To teach programming concepts effectively, educators need methods that enhance visual understanding and student interest. While game-based teaching tools help in this, most focus on computational thinking through gameplay, often neglecting in-depth code understanding and practical skills.

Addressing the identified gap, this poster introduces a programming tool rooted in game design. Our tool utilizes the beloved Puyo Puyo[1],a classic tile-matching game—on the MakeCode Arcade platform by Microsoft, optimized for block-based game creation. The game, with its simple rules and popularity among children, becomes the medium for instructive coding challenges. As they navigate through Puyo Puyo's mechanics, students not only gain proficiency in programming concepts but also enhance their computational thinking.

Our educational tool can be divided into four layers (Fig. 1):

Platform Layer: The foundation of our educational tool is the MakeCode Arcade platform, which is an accessible and robust block-based programming platform specially designed to for beginners.

Project Layer: Utilizing the MakeCode Arcade platform, we've designed a game project based on the child-favorite, Puyo Puyo. Its straightforward rules offer an ideal backdrop for education. Each game feature corresponds to a distinct programming concept, allowing students to apply theory in real-time. By integrating the correct code segments, students progressively assemble the complete game.

Textbook Layer: We designed a textbook layer with code puzzles directly tied to Puyo Puyo game functions. Each puzzle imparts specific programming knowledge, enabling students to grasp game feature implementation and deepen their understanding of programming concepts. This structured approach ensures continuous and layered learning.

Teaching Layer: Our teaching layer, complementing the textbook, uses interactive methods like quizzes and open discussions, promoting understanding and curiosity. By allowing students to experiment with and modify code, they become active learners, fostering an environment of exploration and innovation.

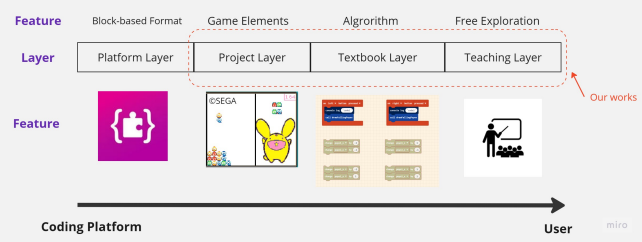


Figure 1: Structure of the Proposed Tool

Our educational tool consists of eight well-structured chapters. Each one focuses on a specific programming concept. These concepts are applied to the Puyo Puyo game, demonstrating their practical use. By inserting correct code snippets, students engage in hands-on learning while developing the game's functions. The chapters are designed for a progressive learning experience. Consequently, students build their understanding as they advance.

2 APPLICATION

2.1 Workshop

Our team conducted a practical workshop in Japan using the proposed educational tool. A group of 12 students between the ages of 9 and 13 attended the workshop. The workshop was organized into four well-structured sessions. Each section was two hours, to permit a thorough exploration of the programming concepts and provide hands-on experience with the tool.

First, the instructor referenced the essential programming concepts pertinent to each section of the textbook. After a comprehensive introduction, students were encouraged to independently grapple with code puzzles related to the recently presented concepts, facilitating active applications of newly learned knowledge.

Once the students attempted the puzzles, the teacher explained the solutions and addressed any raised questions or areas of uncertainty. This was followed by a phase of guided experimentation, where students were given the freedom to explore alternative coding approaches and modify to the game rules and behaviors. This free exploration part underlined the creative aspects of programming and allowed students to gain a more profound understanding of the relationship between the code and its manifestation in the game's functionality.

To evaluate the effectiveness of the workshop and the tool, we adopted a two-pronged assessment approach. First, a ten-question quiz was administered before and after the workshop. The quiz addressed various programming concepts and facilitated a direct measure of the students' progress and comprehension of programming principles. Second, a post-workshop questionnaire was distributed to gain insights into the students' experiences and perspectives.

2.2 Results

The questionnaires highlighted the effectiveness and appeal of our teaching tool. All participants reported a positive workshop experience and voiced their interest in using the game-based approach for learning programming. Importantly, the theme of the game Puyo Puyo seemed to resonate with the students. All the participants were familiar with the game and 58.3% had even played it before.

This pre-existing familiarity and interest in the game potentially heightened their engagement with our teaching tool. Many respondents(81.8%) expressed a high level of interest in the game-based method of teaching. Moreover, all the students indicated a willingness to participate in a similar workshop in the future.

These findings strongly suggest that our proposed tool successfully piques students' interest and facilitates a fun, yet educational, programming learning experience. It underscores the effectiveness of incorporating familiar and enjoyable themes like Puyo Puyo in educational tools to enhance student engagement and learning.

The quiz results showed students' performance improved post-workshop, with average scores rising from 3.1 to 5.3 out of 10. A t-test was performed on the pre- and post-workshop scores, given their normal distribution. The null hypothesis (H0) posited no significant score difference, while the alternative hypothesis (H1) proposed a significant improvement. With a p-value of 0.065 and considering the small sample size, there's strong evidence supporting H1, indicating the tool's effectiveness in enhancing programming skills.

We analyzed students' performance on specific programming concepts and computational thinking abilities using a predefined rubric[2]. The rubric covered programming constructs like sequence, variable, and function, along with computational thinking aspects such as abstraction and decomposition. Each quiz question was mapped to multiple rubric facets, and a correct answer accrued scores from all linked criteria. We computed the highest score each student achieved for individual rubric elements across the quiz, followed by the group's average for each item.

Students showed improvement across all areas, with significant progress in decomposition. This growth reflects our tool's approach of breaking games into smaller functions, facilitating comprehension and problem-solving. The marked improvement, especially in decomposition, underscores our tool's efficacy in comprehensive programming education.

3 CONCLUSION

This poster presents a unique game-design–based programming tool inspired by Puyo Puyo. Unlike conventional tools, it integrates game design, block-based coding, and emphasizes code reading. A workshop with students aged 9–13 showcased its efficacy, with marked improvements in programming skills, computational thinking, and student engagement. Future adaptations will introduce pre-learning content and emphasize student-centered interactive learning.

4 ACKNOWLEDGMENTS

We would like to thank SEGA and Mariko Tamura from D2CR for assisting. We would like to thank SEGA for the official permission of using Puyo Puyo.

REFERENCES

[1] Tamer Koh. 2013. Here's How You Play Puyo Puyo Folks. Video. Retrieved June 25, 2013 from https://www.youtube.com/watch?v=0fyUIuJAJCo

[2] Daisuke Saito, Hironori Washizaki, Yoshiaki Fukazawa, Mariko Tamura, and Yuki Sakuragi. 2019. Rubric to evaluate programming learning of elementary school students. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 1280–1280.

Using Computational Thinking to Learn Creative Writing and Performance Arts: summer experiences of story creation projects

Ruth Torres Castillo
New Mexico State University
Las Cruces, New Mexico, USA
rutorres@nmsu.edu

Tomas Valdez
New Mexico State University
Las Cruces, New Mexico, USA
tvaldez1@nmsu.edu

Enrico Pontelli
New Mexico State University
Las Cruces, New Mexico, USA
epontell@nmsu.edu

Horacio Gonzalez
New Mexico State University
Las Cruces, New Mexico, USA
horaciog@nmsu.edu

Erick Nevarez
New Mexico State University
Las Cruces, New Mexico, USA
erickn02@nmsu.edu

Raena Cota
New Mexico State University
Las Cruces, New Mexico, USA
raenac@nmsu.edu

ABSTRACT

In our modern society, technology has profoundly reshaped various aspects of our lives, influencing the way we work, communicate, and interact. This raises a fundamental question: Is the current K-12 education system adequately preparing students for success in this technology-driven era, where innovation and problem-solving skills are essential? A recent study investigated this inquiry. The goal was to integrate Computational Thinking (CT) into the domains of Creative Writing and Creative Media and measure its impacts. This poster describes the design and refinement of summer camp experiences in which predominantly Hispanic female students participated in hands-on CT-based learning to create their own creative stories and films. Educational outcomes were evident through increased student engagement, enhanced collaboration, and improved problem-solving skills, underscoring the intervention's advantages. Furthermore, the researchers gained transforming insights, recognizing the versatility of CT across domains. Camp participants experienced a shift in perspective, with a greater interest in vocations and domains connected to computing. We share lessons learned from these summer camps in the hopes that they will inform other researchers and practitioners who are interested in designing and deploying similar experiences. In summary, the study outlines a trajectory where students and educators skillfully navigate the complexities of contemporary paradigms.

ACM Reference Format:
Ruth Torres Castillo, Tomas Valdez, Enrico Pontelli, Horacio Gonzalez, Erick Nevarez, and Raena Cota. 2024. Using Computational Thinking to Learn Creative Writing and Performance Arts: summer experiences of story creation projects. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635619

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635619

1 INTRODUCTION AND BACKGROUND

Computational thinking (CT) promotes a methodical approach to addressing challenges and encourages individuals to formulate solutions in comprehensible terms, suitable for execution by either a computer or another person [11]. In the context of our intricate digital landscape, the principles of CT equip individuals to effectively engage with a wide spectrum of challenges [5, 7]. DEPICT (Discover Computational Thinking through Creative Writing) joins the family of interventions that explore the use of non-computing techniques to expose students to CT [4, 6, 10]. The novelty of DEPICT lies in two aspects. First, DEPICT builds on an area that has not been deeply exploited in the past, i.e., the combination of creative writing (CW) and the production of movies/plays. The appeal of this area comes from its popularity among students, especially women, and Hispanics, and the high level of self-efficacy demonstrated by students in the corresponding courses. The second novelty comes from using this creative domain as a target for the infusion of CT - i.e., we use CT as a methodology to teach the creative domain, contending that relevant concepts of CT are already present in the domain, and they simply need to be extrapolated. This paper presents the conception and execution of two DEPICT summer camps. These initiatives collectively engaged high school students, immersing them in hands-on learning experiences deeply rooted in CT. To accommodate the constraints of page limitations, we use this document to share a summary of findings, we provide additional information and insights on our poster presentation.

2 OVERVIEW OF DEPICT SUMMER CAMPS

During the summers of 2022 and 2023, we hosted two two-week camps where 20 local HS students (14 females, 5 males, and one non-binary) participated in an experience centered around Performance and Creative Writing (PCW), with a project-focused emphasis. Throughout the camps, we delved into the worlds of CS, CT, CW, Creative Media (CM), and animation content, using tools like Merge Cubes[3], Specdrums[8], Rocketbooks, and Finch Robots[9]. We introduced new software such as PhotoPea and Capcut. Other web browser applications that were also introduced were CoSpaces[2], Studio Binder[1], and Teachable Machine from Google. During these summer intensives, students learned how to bring CS, CT, CM, and PCW together, developing skills to create remarkable projects. During the 2023 DEPICT summer camp, a "Mystery" concept was

applied to each exercise which aided in collaboration between CS, CM, and PCW. Students applied combinations of traditional and digital techniques to solve riddles, generate whodunit stories, and create imagery intended to deceive their audience and leave them in suspense. Students enjoyed projects with varied combinations of visual and auditory applications, such as a movie poster with voiceover narration and mystery stories coded and presented using Co-Spaces and Merge Cubes. At the camps' finale, students showcased their projects to their families through a Photopea poster design for their Story-Movie, alongside a video, or a Script from Studio Binder.

2.1 Camps collection of data

Retrospective Surveys: DEPICT employed retrospective online surveys to assess the impact and extent of exposure to CT concepts during the duration of the summer camp. Each set of items was asked twice at the end of the program, with one set of questions focusing on how a participant felt "before the program" and another set emphasizing how they feel "after the program".

Pre and post-performative interviews: On the first and last days of the DEPICT camps, each camper was asked specific questions aloud by staff working at the camp. Each interview was audio recorded. Students were asked to describe awareness of CT terms, define the term, and complete tasks that related to loops, decomposition, and abstraction [4].

Camper's Projects: Each camper worked on individual projects: during week 1, campers were introduced to all the technology and CT concepts and during week 2, they worked to create a final project. We encouraged campers to brainstorm personally and socially relevant projects for their final presentation.

2.2 Camps Outcomes

DEPICT summer camps participants experienced modest, positive gains in belonging/identity in STEM, and future plans with computing/technology across items; their retrospective survey scores saw less change in their perceived value of computing and CT. DEPICT summer camp students experienced modest, positive gains in conceptual understanding of CT according to assessment scores from performative interviews. Youth-focused approaches used in DEPICT supported interest, and engagement in the program. Practice and play were vital for student computing identity development and easing uncertainty that students could participate in computing.

3 LESSONS LEARNED

Benefit from parallelism: The parallels between CT and the creative process are apparent, while the differing perspectives appeal to varied learning styles. Every activity used by DEPICT summer camps provided the opportunity to discover how CT was embedded in the host discipline content.

Use a common theme: Applying a common theme in activities assists in collaboration between different disciplines. Using the "Mystery" team for the summer camp helped students build the relationship between CT and CM content.

Learn by experimenting: Experimentation is beneficial upon the initial introduction of CT concepts to those outside of CS. During the 2023 DEPICT Summer Camp, for example, Algorithmic Thinking was used to introduce concepts of character design when Pattern Recognition proved to be a more appropriate CT concept in hindsight.

Use interdisciplinary experts: Combining the expertise of interdisciplinary areas was beneficial to the development of the curriculum. Each of the participants in the design of activities had something to add to each lesson plan. HS teachers added their content knowledge, CS fellows helped by mentoring the activities, and the implementation team helped overview the general process.

Merge CT with pedagogical approaches: CT concepts can enhance previously applied pedagogical approaches in creative classrooms by providing structure and a scientific, methodological approach. We predict that in-service teachers can apply the strategies used during summer camps to foster their future lessons.

4 CONTRIBUTIONS AND FUTURE WORK

By infusing CT principles into CW and film production, DEPICT manages to bridge disciplines, enhancing problem-solving skills, engagement, and collaboration among participants. The outcomes of this project demonstrate that CT has the potential to transcend its conventional boundaries, proving itself as a powerful tool in navigating modern challenges. DEPICT's innovative approach of merging CT with CW offers a distinctive path to engage underrepresented groups, contributing to broader efforts aimed at bridging gaps in computing education and fostering inclusivity. Building upon the benefits of DEPICT, future work could explore the scalability of this approach across diverse educational settings and disciplines. Further research can delve into refining and expanding the curriculum, encompassing a broader range of creative domains and computational concepts. Long-term studies could track the sustained impact of CT exposure on students' academic trajectories, career choices, and problem-solving abilities.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant #2137581.

REFERENCES

[1] Simona Clapan. 2014. web-based filmmaking software. Retrieved October 13, 2023 from https://www.studiobinder.com
[2] CoSpaces.io. 2012. Cospaces Edu for kid-friendly 3D creation and coding. Retrieved October 13, 2023 from https://cospaces.io/edu/
[3] Merge Labs. 2013. Learn science, master stem, be future ready.: AR/VR Learning Creation. Retrieved October 13, 2023 from https://mergeedu.com/cube
[4] Carolyn Lamb, Daniel G Brown, and Charles LA Clarke. 2018. Evaluating computational creativity: An interdisciplinary tutorial. *ACM Computing Surveys (CSUR)* 51, 2 (2018), 1–34.
[5] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. Computational thinking for youth in practice. *Acm Inroads* 2, 1 (2011), 32–37.
[6] Dr Mahsa Mohaghegh and Michael McCauley. 2016. Computational thinking: The skill set of the 21st century. (2016).
[7] L. Pollock, C. Mouza, K. R. Guidry, and K. Pusecker. 2019. Infusing computational thinking across disciplines: Reflections & lessons learned. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 435–441.
[8] Sphero. 2018. Specdrums. Retrieved October 13, 2023 from https://sphero.com/collections/all/family_specdrums
[9] BirdBrain Technologies. 2020. Finch Robot 2.0. Retrieved October 13, 2023 from https://www.birdbraintechnologies.com/products/finch-robot-2-0/
[10] K. Tengler, O. Kastner-Hauler, and B. Sabitzer. 2021. Enhancing Computational Thinking Skills using Robots and Digital Storytelling.. In *CSEDU (1)*. 157–164.
[11] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

Assessing the Efficacy of Goal-Based Scenarios in Scaling AI Literacy for Non-Technical Learners

Ying-Jui Tseng
Carnegie Mellon University
Pittsburgh, USA
yingjuit@andrew.cmu.edu

Ruiwei Xiao
Carnegie Mellon University
Pittsburgh, USA
ruiweix@andrew.cmu.edu

Christopher Bogart
Carnegie Mellon University
Pittsburgh, USA
cbogart@andrew.cmu.edu

Jaromir Savelka
Carnegie Mellon University
Pittsburgh, USA
jsavelka@andrew.cmu.edu

Majd Sakr
Carnegie Mellon University
Pittsburgh, USA
msakr@cs.cmu.edu

ABSTRACT

AI's pervasive role in various fields highlights the imperative for the workforce to adeptly leverage its potential. While numerous courses cater to developers, there exists a discernible void for the wider community of AI users. To address this, our study introduces the 'AI User' course—a suite of interactive modules hosted on the Sail() platform, designed specifically for non-technical individuals utilizing Goal-Based Scenario (GBS) learning. We conducted a controlled experiment to ascertain whether GBS offers superior learning gains in AI literacy compared to traditional deliberate practice using multiple choice questions.

ACM Reference Format:
Ying-Jui Tseng, Ruiwei Xiao, Christopher Bogart, Jaromir Savelka, and Majd Sakr. 2024. Assessing the Efficacy of Goal-Based Scenarios in Scaling AI Literacy for Non-Technical Learners . In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635620

1 INTRODUCTION

AI's growing presence in diverse sectors, such as voice assistants, facial recognition, and recommendation systems, underscores its role in people's everyday lives as well as in diverse professional settings. Yet, its full potential can only be harnessed when used by operators who understand its capabilities and limitations.

While many AI introductory courses are tailored for developers and researchers, few are aimed at users of AI. It is crucial for users to understand how AI tools function based on their training data and how, e.g., environment changes can influence a system's performance.

We aim to design, develop, and evaluate innovative methods and tools for remote, asynchronous, at-scale learning, with the goal of training learners to make informed, effective, judicious use of the AI-enabled features of technology that are part of the modern workplace, and to advocate for effective use of AI. A central concern for this remote, intensive online training is ensuring active engagement. To tackle this, we have adopted several engagement techniques, emphasizing Goal-Based Scenario learning (GBS) [1].

In this study, we assess the 'feature engineering' module within our 'AI User' course, which is delivered through a simulation environment on the `Sail()` platform[1]. Through a controlled experiment, we aim to determine whether Goal-Based Scenarios (GBS) result in greater learning gains in AI literacy compared to deliberate practice using multiple choice questions. By comparing the learning gain in both approaches, we aim to answer the following research question: Are scenario-based activities more effective than traditional learning activities (e.g. multiple-choice questions) for learning goals in AI literacy?

2 METHOD

2.1 Goal-Based Scenario

In our AI User course, each project revolves around an authentic scenario motivating all learning activities. For the feature engineering project, the following scenario provides real-world context and aligns with our learning objectives:

> Faced with escalating customer churn rates, your company endeavors to craft a targeted marketing campaign to retain customers nearing the end of their contracts. Collaborating with the data team, the aim is to devise a predictive model for customer churn, ensuring the timely reach of promotions to those most at risk.

Students assume the role of a non-technical telecom company marketing manager tasked with leveraging AI to mitigate customer attrition. For context, an open-source telecom customer churn data set is incorporated throughout the project.

2.2 Example Learning Activities

In this project, we designed four learning activities. This section will delve into two of these activities and the specific learning objectives they aim to address.

Activity 2: Feature Combination for Churn Prediction This activity targeted the learning objective: *LO2: Analyze and juxtapose*

[1] https://sailplatform.org/

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635620

the performance of different iterations of the same machine learning system that utilizes varying feature combinations. Learners are asked to collaborate with the data team to develop a customer churn prediction model. Given 20 types of customer data, they must select the six most pertinent types. Model training is then simulated, and the system provides accuracy feedback. Students are presented with a coefficient table as feedback to refine their feature selections.

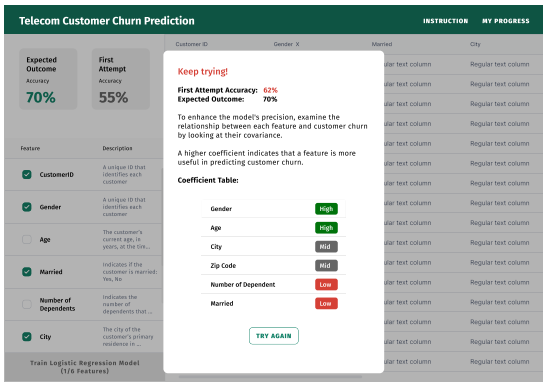


Figure 1: Activity 2

Activity 3: Model Comparison - Simple vs. Advanced This activity targets the learning objective: *LO3: Contrast the precision of basic machine learning systems against more sophisticated counterparts.* Learners are given the following scenario:

> In order to decrease the customer churn rate, you decide to ask the Data Science Team to build a customer churn prediction model. This will allow the marketing team to target customers that are likely to churn and send them personalized offerings. The Data Science lead tells you they have two models that could achieve this goal

Initially, learners engage with a limited data set, observing the efficacy and transparency of simpler models. Following this, a larger data set showcases the nuanced performance improvements of advanced models.

3 EXPERIMENT

This pilot study adopts a between-participants pre-post test design. We recruited 64 participants with non-technical backgrounds from Prolific [2]. Each participant was tasked with sequentially completing a pre-test, reading material, a learning module, and a post-test within approximately 45 minutes. Participants were randomly allocated to one of two groups: the experimental group, which utilized the simulation environment described above, and the control group, which aimed to teach the same learning objectives by multiple-choice questions. Both tests consisted of identical sets of 16 multiple-choice questions, aligned with the intended learning objectives. The difference in scores between these two tests was used to measure learning gains.

[2]Prolific is an online platform for participant recruitment. https://app.prolific.com/

4 ANALYSIS AND RESULTS

4.1 Quantitative Analysis

The experiment group achieved a higher learning gain in overall performance (Table 1, C: Control, E: Experiment). However, in terms of each participant's learning gain, there is no statistically significant difference (p=0.9236>0.05) between groups in any types of the questions.

	Overall	LO2.2	LO3.1	LO3.2
C	1.81%	-6.2%	-4.2%	-10.3%
E	**2**.41%	-17.2%	-7.4%	**3**0.1%

Table 1: Learning Gain on Different Types of Questions

4.2 Qualitative Analysis

While most of the scores in each LO show variant levels of increment after students studying the learning module with no little difference among both groups, we noticed significant decrement on accuracy on LO2.2 and LO3.1, and contrast effects for different groups on LO3.2, where the accuracy decreased in the group using MCQ learning material, and increased in the group using simulation environment (Table 1). To further investigate potential causes, we interviewed participants for evidence of the possible causes.

The causes of negative effects resulted from students' insufficient learning gain. Students tend to remember the concept, but fail to master the context and conditions that apply to the concept. For example, in LO3.1, when asked "When comparing of a simple machine learning system to an advanced one, which feature primarily influences the accuracy difference?", more students chose an incorrect answer "process speed" in the post-test than pre-test, because they were largely influenced by the training time simulation in the module. The module managed to tell students that speed matters, but failed to help learners to identify the scenarios when they need to prioritize process time. A positive example that demonstrates the success of simulation environment is LO3.2. While the learning gain in the control group decreased 10.3%, it increased 30.1% in the experimental group. This might be due to the scenario activities provided several opportunities to explore various combinations and consequences to help students' knowledge transfer. In summary, future designs should elicit learning with required depth and breadth.

5 CONCLUSION

This study introduces a scalable approach to AI literacy training for non-technical individuals and its evaluation. The results underscore the potency of scenario-based activities in enhancing learning outcomes and suggest an avenue for AI educators to foster deeper comprehension and application of concepts. Building on this, we're designing eight comprehensive AI literacy modules and curating guidelines tailored for AI educators and non-technical students.

REFERENCES

[1] Roger C Schank, Andrew Fano, Benjamin Bell, and Menachem Jona. 1994. The design of goal-based scenarios. *The journal of the learning sciences* 3, 4 (1994), 305–345.

Math IDE: A Platform for Creating with Math

Sierra Wang
sierraw@cs.stanford.edu
Stanford University
CA, USA

John Mitchell
mitchell@cs.stanford.edu
Stanford University
CA, USA

Nick Haber
nhaber@stanford.edu
Stanford University
CA, USA

Chris Piech
piech@cs.stanford.edu
Stanford University
CA, USA

ABSTRACT

To inspire student engagement in middle school math, we explore the possibility of using generative AI to enhance the creativity of math learning. We present the Math IDE, a math education environment in which students learn about math concepts by building artifacts. We aimed to create a platform in which students can engage with mathematical concepts, create an artifact that embodies the math that they are learning about, and practice their high-level specification skills. In the current iteration of the Math IDE, students can create custom web pages by describing and demonstrating understanding of the math that is involved in the web page. In this short overview, we describe our process and discuss several open questions regarding the design and application of this novel method of math education.

ACM Reference Format:
Sierra Wang, John Mitchell, Nick Haber, and Chris Piech. 2024. Math IDE: A Platform for Creating with Math. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635572

1 INTRODUCTION

Empowering students with the ability to create can drive their motivation to learn [2]. We have seen this in computer science education where students are able to create artifacts as they learn how to program. Due to its ability to generate code, generative AI has the potential to bring similarly creative experiences to education in unprecedented ways. In this work, we explore how we can utilize generative AI to bring the act of creating to middle school math, and consequently, increase engagement and motivation in the subject.

2 MATH IDE

We set out to develop a platform where a student can do the following:

(1) Engage with and improve their understanding of mathematical concepts.
(2) Create an artifact that embodies the mathematical concepts that they are using.
(3) Practice their high level specification skills by working with artificial intelligence (AI) based tools.

We call this platform the Math IDE. The goal of this platform is to increase student engagement in middle school math by bringing the act of creating to math practice. Thus, the Math IDE must be an environment in which students feel the freedom to create, yet where generative AI is confined to reasonable tasks.

2.1 What can a student create with math?

The first type of project that a student can create with the Math IDE is a web page that has inputs, an output, and a run button. In the Math IDE, the student describes the inputs and the function that should be used to compute the output. The system utilizes GPT to generate a JavaScript function and a set of sample inputs reflecting the student's description. For each sample input, the student must compute the output that they expect, and compare this with the output computed by the generated function. If the outputs do not match, the student must either modify their output or assert that the generated function must be wrong (in which case, a new function must be generated). Once the student has verified all of the sample inputs, they can design the layout and background of their web page, and share it with their friends. Figure 1 shows this process.

We designed this project to meet the goals outlined in Section 2. The sample inputs are meant to improve and ensure the student's understanding of the math concepts that they are using, while still providing a creative experience. We also hope that through this project, students learn how to describe their ideas with clarity and precision, a skill that is important when using generative AI.

2.2 What is the user interface?

The user interface is inspired by programming IDEs. In a programming IDE, the student is able to run their work often and receive immediate feedback on how they are doing, and share what they build. We believe these qualities improve engagement and motivation, so we aimed to incorporate them in our interface.

We also explored numerous decisions about the feel of the interface. We spoke to multiple experts in math education and identified two distinct methodologies. The first methodology is to make our environment feel as exciting as possible. With this in mind, we designed the environment so that the student feels like they are

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635572

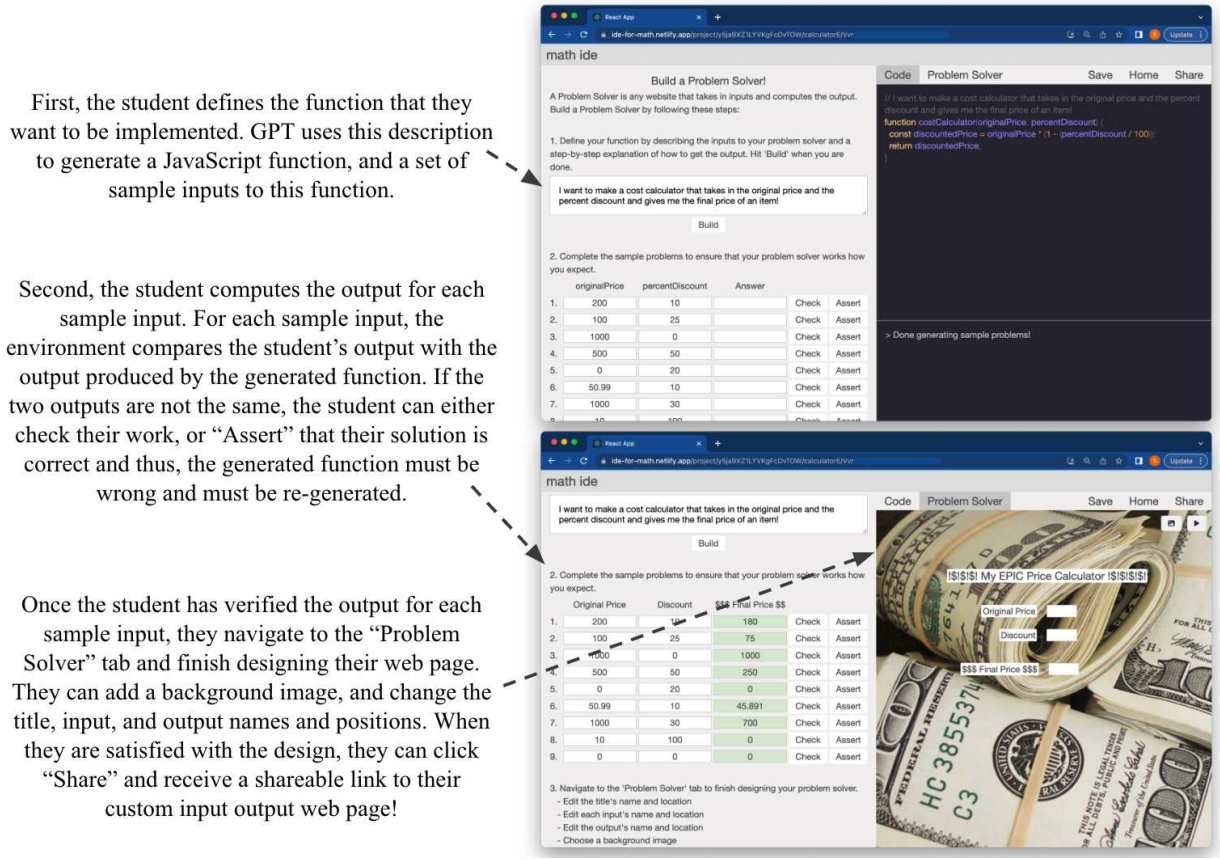


Figure 1: An experimental user interface for the Math IDE.

coding. An example of this can be seen in Figure 1, where we display the generated code in a terminal like window. The second methodology is to make the experience feel familiar so that less confident students are not overwhelmed by the new environment. We created multiple interfaces with this goal, where the language and design is as simple and familiar to the student as possible. We have not yet determined if one of these methodologies is better than the other.

2.3 What are other types of projects?

The most significant challenge in developing this platform is identifying what projects the students can build. We must be able to extrapolate the mathematical aspects of the artifact for the student to define, and then rely on GPT to generate the result reflecting the student's work. Our first type of project, the custom input/output web page, demonstrates the feasibility of this goal. We believe this type of project could be used to teach certain middle school math concepts, like percentages, decimals, and fractions. We hypothesize that other project types would be better suited for teaching other concepts. For example, a game maker where the student defines the physical aspects of the game, GPT generates it, and then the student is able to play their custom game might be better suited for teaching geometry concepts, such as angles and shapes.

2.4 How did we build it?

We built the Math IDE as a React web application. All GPT calls are made to OpenAI's Create Completions API [1]. We experimented with different LLMs, and found that text-davinci-003 is sufficient for our current needs.

3 CONCLUSION

A key remaining question lies in determining where this tool will be positioned, whether within traditional classrooms, online platforms, or a blend of both. To evaluate this tool, we need to assess how effectively it stimulates student engagement and how it impacts the overall learning outcomes of the students. There is also a need to address the security aspect, to ensure GPT is appropriately bounded in its generation. Moving forward necessitates collaborative efforts from educators and curriculum developers to transform this opportunity into an effective approach to learning mathematics.

REFERENCES

[1] 2021. Create Completions - API Reference. https://platform.openai.com/docs/api-reference/completions/create. Accessed: 2023-08-17.
[2] Daniel L Schwartz, Jessica M Tsang, and Kristen P Blair. 2016. *The ABCs of how we learn: 26 scientifically proven approaches, how they work, and when to use them.* WW Norton & Company, Chapter M is for Making | Producing interest and practical knowledge, 153–165.

Measuring CS Student Attitudes Toward Large Language Models

Jason Lee Weber
weberjl@uci.edu
University of California, Irvine
Irvine, California, USA

Barbara Martinez Neda
barbarm@uci.edu
University of California, Irvine
Irvine, California, USA

Kitana Carbajal Juarez
kitanac@uci.edu
University of California, Irvine
Irvine, California, USA

Jennifer Wong-Ma
jwongma@uci.edu
University of California, Irvine
Irvine, California, USA

Sergio Gago-Masague
sgagomas@uci.edu
University of California, Irvine
Irvine, California, USA

Hadar Ziv
ziv@ics.uci.edu
University of California, Irvine
Irvine, California, USA

ABSTRACT

With the mainstream adoption of Large Language Models (LLMs), members of both academia and the media have raised concerns around their impact on student learning and pedagogy. Many students and educators wonder about the pedagogical fit of this emerging technology. We aim to measure the adoption of and attitudes toward LLMs among the CS student population at an R1 University to determine how students are using these new tools. To this end, we conducted a large survey study targeting two populations participating in computing courses at the university: intro-sequence students (ISS) and experienced students (ES).

In our preliminary results from Spring 2023, we've found several significant differences among the views of over 700 respondents across the two groups. Most students reported LLMs' unparalleled potential for quick information access, yet many harbor concerns about the reliability of the LLM responses, and the impact on academic integrity. Additionally, while ES have rapidly integrated LLMs into their learning, ISS remain cautious of the tools, highlighting a stark contrast in adoption rates between the groups.

LLMs are clearly going to reshape pedagogical approaches and student engagement. Our study hopes to provide insight on the nuanced student attitudes toward LLMs. For example, the notable reservations expressed by ISS illustrate an imperative for careful, informed, and ethical integration to ensure these tools enhance rather than compromise the educational experience. In the future, we plan to continue tracking student attitudes in order to gain further understanding of the changing perceptions of LLMs and their impact.

ACM Reference Format:
Jason Lee Weber, Barbara Martinez Neda, Kitana Carbajal Juarez, Jennifer Wong-Ma, Sergio Gago-Masague, and Hadar Ziv. 2024. Measuring CS Student Attitudes Toward Large Language Models. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635604

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635604

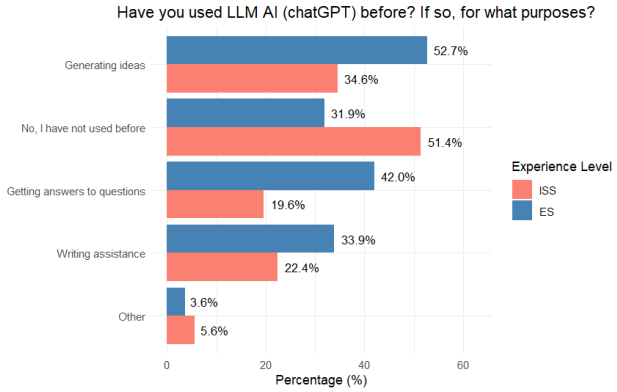
1 INTRODUCTION

Since the release of ChatGPT in November 2022 [1], there has been exponential growth of AI tools and the use of AI. ChatGPT passed 1 million users within a week of launch [2, 5] and is now reporting over 100 million monthly users [7]. This widespread adoption has led to concerns over its impact on education [3]. Often the discussion revolves around academic integrity concerns and the spread of misinformation [4, 6, 8]. Our work focuses on measuring the attitudes of CS towards LLMs and their role in higher education.

To this end, in our R1 institution, we began to survey our CS student population in Spring 2023 on their attitudes and views towards AI LLMs. We plan to continue to survey each term in order to capture the possible change of viewpoints as LLM adoption continues. We pose the following research questions:

- What are the current student views of LLMs as part of their education?
- Do students of different experience levels have different attitudes?
- As LLMs improve, is there a shift in attitude among the students?

Figure 1: Student LLM Usage



2 APPROACH

To gather insights from individuals in the CS student population at our R1 university regarding their attitudes and viewpoints on

LLMs, we began to survey (with IRB approval) intro-sequence students (ISS) and experienced students (ES) in the Spring of 2023. ISS includes all students enrolled in the introductory programming sequence for CS majors at the institution, although enrollment is not limited by major, with many students from outside of CS taking this sequence (or parts of it). ES were all enrolled in computing courses beyond the aforementioned introductory sequence.

The survey collects demographic information and student attitudes around prior usage, general educational benefits, and personal benefits of LLMs in education. Additionally, students are asked about whether they believe LLMs would have a positive or negative impact on academic integrity and if they thought LLM usage should be considered an academic integrity violation.

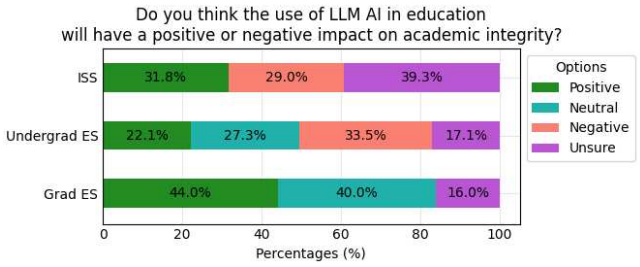
3 PRELIMINARY RESULTS

In Spring 2023, we collected responses from ~100 Intro-Sequence Students (ISS) and ~650 Experienced Students (ES). Our preliminary analysis highlighted the following findings.

In terms of LLM usage, over half of ISS and had not used an LLM yet. However, ES students have consistently tried LLMs (~70%). When we look at graduate ES separately from undergraduate ES, we see that 96% of graduate ES have tried using LLMs. Figure 1 briefly summarizes ways students are utilizing LLMs. ~80% of ES were concerned with the accuracy of answers provided by LLMs. Only ~11% of ES think they will have a difficult time understanding responses provided by the LLM. Over 80% of all students believe that LLMs generally offer faster access to information, yet only ~30% of ISS viewed this as a personal benefit to their learning.

The data also reveals quite a large disparity between groups when it comes to views on the impact that LLMs will have on academic integrity. In Figure 2, 84.0% of graduate ES believe that there will be a positive or neutral impact and none believe it will have a negative impact.

Figure 2: Perceptions of LLM Impact on Academic Integrity



Lastly, due to the ISS survey being a small part of a much larger survey on student success, extra data is available to examine additional correlations that are not available for ES. A notable correlation occurs between students' reported anxiety levels and their use of LLMs. In particular, ISS students who report often or always feeling anxious seem less inclined to use LLMs for things like "Generating ideas" and "Faster access to information".

For example, just 15.63% of students who often or always feel anxious used LLMs for idea generation compared to 42.67% of their less anxious peers. Similarly, when identifying the benefits of LLMs, a small majority of 65.63% perceive "Faster access to information" as a benefit, while 86.67% of their less anxious peers did the same.

4 CONCLUSION

Our study hopes to provides insight on the nuanced student attitudes toward LLMs. Our initial results suggest that, while students recognise the benefits, they harbour concerns about the accuracy of answers given by LLMs. Furthermore, experience-level disparities demonstrate a cautious testing of LLMs, dispelling the myth of swift, widespread adoption among the student population. Lastly, our preliminary results show an interesting correlation between student anxiety levels and LLM usage, suggesting that students with higher levels of anxiety may be less willing to adopt than their less anxious peers.

Moving forward, we plan to advance our research in the following directions:

(1) **Continued Data Collection:** We will continue our data collection efforts each term so that we may examine changes (or a lack thereof) in student attitudes as LLM adoption continues. Additionally, we will expand the survey to more R1 universities.
(2) **Student Interviews:** We plan to conduct semi-structured interviews with ISS and ES in order to gain a deeper understanding of the reasoning behind their views and attitudes.
(3) **Statistical Analysis:** To test the statistical significance and strength of the correlations we find, we will conduct χ^2 tests of independence and calculate Kramer's V values.

REFERENCES

[1] 2022. Introducing ChatGPT. https://openai.com/blog/chatgpt. Accessed: August 17, 2023.
[2] Katharina Buchholz. 2023. Infographic: Threads Shoots Past One Million User Mark at Lightning Speed — statista.com. https://www.statista.com/chart/29174/time-to-one-million-users/. Accessed: August 19, 2023.
[3] Cindy Gordon. 2023. *How Are Educators Reacting To Chat GPT?* https://www.forbes.com/sites/cindygordon/2023/04/30/how-are-educators-reacting-to-chat-gpt/ Accessed: August 18, 2023.
[4] Will Douglas Heavenarchive. 2023. ChatGPT is going to change education, not destroy it. https://www.technologyreview.com/2023/04/06/1071059/chatgpt-change-not-destroy-education-openai/
[5] Krystal Hu. 2023. *ChatGPT Sets Record for Fastest-Growing User Base - Analyst.* https://www.reuters.com/technology/chatgpt-sets-recordfastest-growing-user-base-analyst-note-2023-02-01/ Accessed: March 23, 2023.
[6] Kathryn Hulick. 2023. How ChatGPT and similar AI will disrupt education. https://www.sciencenews.org/article/chatgpt-ai-artificial-intelligence-education-cheating-accuracy
[7] Dan Milmo. 2023. ChatGPT reaches 100 million users two months after launch — theguardian.com. https://www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app. Accessed: August 19, 2023.
[8] Kevin Roose. 2023. Don't Ban ChatGPT in Schools. Teach With It. https://www.nytimes.com/2023/01/12/technology/chatgpt-schools-teachers.html

Computational Thinking in STEM Teaching: Preservice Teachers' Conceptualizations and Practices

Wei Wei
University of California, Los Angeles
weiwei17@ucla.edu

Jesse Ha
Montclair State University
haj@montclair.edu

Kimberley Gomez
University of California, Los Angeles
kimgomez@ucla.edu

Anuradha Ghosh
Montclair State University
ghosha3@montclair.edu

ABSTRACT

Within a two-year urban teacher residency program at [University], which focuses on training high school teachers to incorporate computational thinking (CT) into STEM education with an emphasis on equity, this study delves into how these preservice teaching residents (TRs) understood and applied CT in high school STEM settings. Based on our initial thematic content analysis of ten 30-minute artifact-based interviews, teachers' understanding and application of CT could be categorized in two ways: 1) One group of teachers showcased clear, explicit knowledge of CT and actively incorporated it into their instruction; 2) Another group, despite their evident understanding of CT, subtly and implicitly integrated CT into their teaching methods. Our findings suggest that while TRs may differ in whether they explicitly or implicitly integrated CT into their lesson plans, their comments suggest that they have foundational understandings and hold values that suggest they can build upon them. The TRs held definitional understandings of CT and had intentions to design and teach with CT integration, explicitly, or more commonly, implicitly. All of the TRs recognized opportunities in their lesson plans for teaching CT to students, which suggests the potential for more explicit integration in the future. These insights guide our subsequent efforts in teacher training, ensuring educators are equipped to weave CT into STEM lessons. Our findings contribute to a growing body of work that seeks to build strong preservice teaching practices, as they learn to translate CT knowledge into effective teaching practices while working towards higher levels of CT integration.

CCS Concepts

Social and Professional Topics • Professional Topics • Computing Education • Computational Thinking

KEYWORDS: Teacher Education, Computational Thinking, Preservice Teachers, Equity, STEM

*

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE'24, March 20-23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03. https://doi.org/10.1145/3626253.3635586

ACM Reference format:

Wei Wei, Jesse Ha, Kimberely Gomez and Anuradha Ghosh. 2024. Computational Thinking in STEM Teaching: Preservice Teachers' Conceptualizations and Practices. *In Proceedings of 2024 ACM Conference on Computer Science Education (SIGCSE'24), March 20-23, 2024, Portland, OR, USA*, ACM, New York, NY, USA. 2 pages. https://doi.org/10.1145/3626253.3635586

1 THEORETICAL FRAMEWORK

STEM subjects have predominantly been leveraged to teach coding or computer science [5]. Building on the insights of Yadav [5], Hemmendinger [4], and others, this paper presents a perspective emphasizing the benefits of embedding Computational Thinking (CT) within STEM education. This not only augments discipline-specific learning but also fosters CT proficiency. For example, Yadav et al. [5] view CT as a technique for problem-solving, encompassing the decomposition of problems, devising solutions, evaluating their adaptability, and deciding when to utilize computer resources. Similarly, Hemmendinger [4] posits that CT education strives to empower people with the means for applying computational to problem-solving, spurring inventive thinking in various areas such as economics, physics, arts, among others. STEM education naturally offers a conducive environment for students to grasp and employ CT concepts. Concrete studies have demonstrated that the deliberate inclusion of CT in STEM substantially amplifies learning in STEM topics across elementary, middle, and high school educational stages [1][2][3]. Nonetheless, there remains a gap of how to instill a deep conceptual and skill-based understanding regarding CT integration among preservice teachers.

2 METHODS

2.1 Study Context

The research was conducted within an urban teacher residency at [University], which is dedicated to training math and science educators who are committed to promoting equity. These educators are prepared to serve the most marginalized high school students in public school districts on the west coast. This initiative adopts a two-year cohort structure. During the first year, Teaching Residents (TRs) undertake master's degree courses covering teaching techniques, coding expertise (specifically with R studio),

and specialized training sessions on Computational Thinking (CT) and equity principles. Concurrently, these TRs begin their student teaching phases, mentored by experienced teaching partners. In the subsequent year, TRs independently take on full-time teaching roles while concurrently working on their master's culminating project.

2.2 Participants

The participants of our research involved 10 TRs out of a total of 16 from this program's second cohort - comprised of six math educators and four science educators. While we reached out to the entire group of 16 TRs for participation, only 10 responded to our interview invitation. The demographic characteristics (such as educational background and ethnicity) of the participating TRs in our study align well with the broader group of 16 TRs. The participants we interviewed embarked on the program in the academic year 2021-2022. By the time of our interviews, they had successfully completed their master's coursework. Currently, they are engaged in full-time teaching roles at public high schools, primarily catering to an urban, low-income, Latinx student population.

3 DATA SOURCES AND ANALYSIS

Data was gathered through 30-minute artifact-based interviews on Zoom with each of the 10 TRs in early 2023. In these sessions, educators elaborated on and evaluated their self-constructed lesson plans that merged CT with either math or science. The interviews, while structured, allowed for supplementary inquiries. These inquiries delved into the educators' underlying reasons for the structure of their lesson plans, their pedagogical choices, and their objectives for equality in the context of CT incorporation.

The interview data was analyzed via content thematic analysis. Initially, we discerned the TRs' understanding of CT integration and their teaching strategies. During our preliminary review and coding of the interviews, we used an in vivo inductive method, drawing codes directly from the data. Subsequently, we deductively formulated a codebook pinpointing primary codes that encapsulated the TRs' notions of CT and their integration strategies. We then revisited and recoded the interviews guided by this refined codebook. The results presented here encompass the preliminary categories and motifs unearthed through our scrutiny.

4 PRELIMINARY FINDINGS

From our initial analysis, we discerned two distinct teacher profiles in relation to their understanding and application of CT integration. Firstly, a subset of the 10 teachers could articulate precise definitions of CT and highlight the CT skills embedded in their lessons. In contrast, the second set of teachers, while exhibiting a clear grasp of CT, integrated it more subtly into their instructional methods based on their lesson plan discussions and remarks.

Before diving deeper into these two teacher categories, it's essential to elucidate our interpretations of "explicit knowledge of CT" versus "implicit knowledge of CT," as well as the distinctions between "integrating CT explicitly" and "integrating CT implicitly." In this context, "explicit knowledge of CT" means that teachers can outline CT definitions encompassing its broad and commonly accepted facets. They can proficiently articulate CT skills such as problem-solving and abstraction. Conversely, "implicit knowledge of CT" implies situations where teachers understand CT's essence but might not convey its principles and related skills as lucidly. In terms of CT application, "integrating CT explicitly" pertains to methods like using CT-specific terminology during instruction, prioritizing CT when formulating lesson plans, among other direct strategies. In contrast, "integrating CT implicitly" denotes instructional methods that intrinsically foster CT learning, even if teachers don't intentionally present students with overt opportunities to acquire and practice CT abilities.

While our analysis is in its initial stages, the data points toward an intriguing trend. Even though there were variations in how explicitly teachers incorporated CT into their lessons, the foundational elements for such integration were evident even in the nascent phases of their teaching journey. Teachers possessed a clear conceptual understanding of CT and were inclined to integrate CT into their lesson designs and teaching, be it explicitly or, as was more typical for this group, implicitly. Each teacher identified moments within their lesson plans that lent themselves to CT instruction, hinting at the possibility of more direct integration in future lessons.

ACKNOWLEDGMENTS

This work is funded by the U.S. Department of Education Award # U336S190038.

REFERENCES

[1] Valerie Barr and Chris Stephenson, 2011. Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1), 48–54. DOI:https://doi.org/10.1145/1929887.1929905

[2] Li Cheng, Xiaoman Wang, Albert D. Ritzhaupt, 2022. The effects of computational thinking integration in STEM on students' learning performance in K-12 education: A meta-analysis. *Journal of Educational Computing Research*, 61(2), 416-443. DOI:https://doi.org/10.1177/07356331221114183

[3] Shuchi Grover and Roy Pea, 2013. Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, 42(1), 38–43. DOI:https://doi.org/10.3102/0013189X12463051

[4] David Hemmendinger, 2010. A please for modesty. *ACM Inroads*, 1(2), 4–7. DOI:https://doi.org/10.1145/1805724.1805725

[5] Aman Yadav, Hai Hong, and Chris Stephenson, 2016. Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms, *TechTrends*, 60(6), 565-568. DOI:https://doi.org/10.1007/s11528-016-0087-7

Perfect Snake: Exploring Algorithms Through the Snake Game

William Weinberger
wlweinbe@usc.edu
University of Southern California
Los Angeles, CA, USA

Jingsai Liang
jliang@westminsteru.edu
Westminster University
Salt Lake City, UT, USA

Xi Chen
xi.chen@uvu.edu
Utah Valley University
Orem, UT, USA

ABSTRACT

Applying various algorithms learned from the courses of Algorithms and Artificial Intelligence to solve real-world problems is intriguing, especially considering that most problems found in textbooks tend to be either trivial, theoretical, or oversimplified. In this poster, we present a challenging project where we apply a variety of common algorithms and their variations to solve the snake game. Our project's primary goal is to develop a *perfect* snake algorithm, defined as one that collects food until the grid is entirely covered by the snake, indicating the completion of the game, ideally achieved in as few steps as possible. This project is particularly intriguing as it offers a novel approach to completing the game, a task that humans often find challenging.

We have explored a range of algorithms, including basic graph searching algorithms like BFS, advanced searching algorithms such as A*, modern techniques like deep Q learning, and a Hamiltonian circuit-based algorithm. Given the complexity of this problem, we have made modifications to most of the standard algorithms, utilizing both brute force and human intervention. Our aim is not only to demonstrate the feasibility of these algorithms but also to reach the exciting milestone of creating a perfect snake on a large grid, surpassing human capabilities.

In summary, this project has not only deepened the student's understanding of algorithms and enhanced the student's coding skills, but it has also exposed the student to the complex challenges faced in real-world applications.

ACM Reference Format:
William Weinberger, Jingsai Liang, and Xi Chen. 2024. Perfect Snake: Exploring Algorithms Through the Snake Game. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635597

1 INTRODUCTION

The snake game is a classic video game in which players strategically maneuver a snake to collect as much food as possible before the game ends. Attempts to complete this game using computer agents have been made in the past, employing three primary methods: Hamiltonian circuit, graph searching techniques like A* algorithm, and deep learning methods like Deep Q-Learning [1]. However, achieving a perfect snake on a large grid, where the snake covers the entire grid using as few steps as possible, presents a challenge beyond human capabilities.

This intriguing problem can be partially solved by applying algorithms learned from courses in Algorithms and Artificial Intelligence, with certain adaptations involving brute force and human intervention. We explored a range of algorithms, including BFS (Breadth-first search), Backtracking, A*, Deep Q-Learning, and the Hamiltonian circuit. We will explain the details of two of these algorithms in Section 2 and present our results and discussions in Section 3.

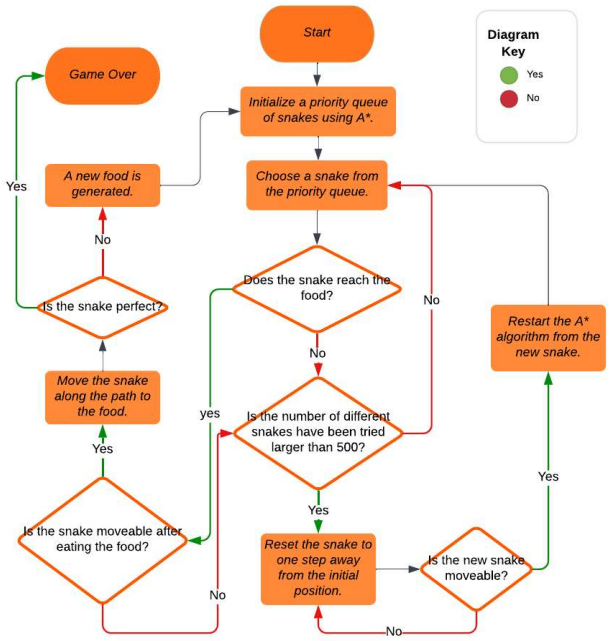


Figure 1: Enhanced A* Algorithm

2 ALGORITHMS

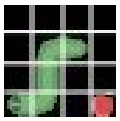
2.1 Enhanced A* Algorithm

Figure 1 depicts a flowchart of an enhanced A* algorithm with two significant modifications compared to the standard A* algorithm. The first modification involves setting a maximum number of loops for A* in cases where it cannot find a solution within a reasonable time. The second modification ensures that the solution found by A* results in the snake's survival after eating the food.

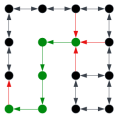
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03
https://doi.org/10.1145/3626253.3635597

	Perfect Snake	Succeess Rate	Shortest Path	Maximum Size	Time Complexity
Enhanced Hamiltonian Circuit	Yes	100%	Yes	4*4	At Least Exponential
Enhanced A*	Possible	59% on 8*8 25% on 20*20	No	30*30	At Least Exponential
Enhanced BFS	No	N/A	Yes	16*16	At Least Exponential
Enhanced Backtracking	No	N/A	No	16*16	At Least Exponential
Deep Q-Learning	No	N/A	No	N/A	N/A

Table 1: Summarization of Explored Algorithms



(a) Snake on 4*4 Grid



(b) Corresponding Directed Graph

Figure 2: Convert Snake and Grid into a Directed Graph

2.2 Enhanced Hamiltonian Circuit Algorithm

A Hamiltonian circuit on a graph is a path that visits all vertices once and returns to the starting vertex. While this method may seem perfect for solving the perfect snake game since, in theory, the snake will eventually reach the food without dying permanently if it follows the Hamiltonian circuit. However, it contradicts the goal of the snake moving along the shortest path to reach the food.

Our combined approach achieves the shortest path by employing BFS while keeping the graph as Hamiltonian, thereby preventing the snake from dying after eating the food. The flowchart of this enhanced Hamiltonian circuit algorithm is depicted in Figure 3, following the conversion of the snake and the grid into a directed graph, as illustrated in Figure 2.

3 RESULTS AND DISCUSSIONS

The algorithms we've explored are summarized in Table 1. While the Hamiltonian algorithm is theoretically the best, it's limited to a 4x4 grid due to its NP-hard nature. A more practical alternative, although it doesn't guarantee the shortest path, is the A* algorithm, which can handle grid sizes up to 30x30. You can view four screenshots of the perfect snake on an 8x8 grid in Figure 4. Video demos of the perfect snake created by our algorithms are available at the following URLs: https://youtu.be/6O6-htzu1TA (16x16) and https://youtu.be/LZKPF8cpuCE (8x8).

The time required by almost all algorithms in Table 1 grows at least exponentially, limiting our ability to create a perfect snake on even larger grids. However, with a 59% success rate on an 8x8 grid using the A* algorithm, the agent we developed in this project notably outperforms most humans.

The student, who participated in this exploration project in a capstone class after taking algorithm and AI courses under the mentor's supervision, gained valuable hands-on experience in understanding algorithms learned in class and coding at an advanced level.

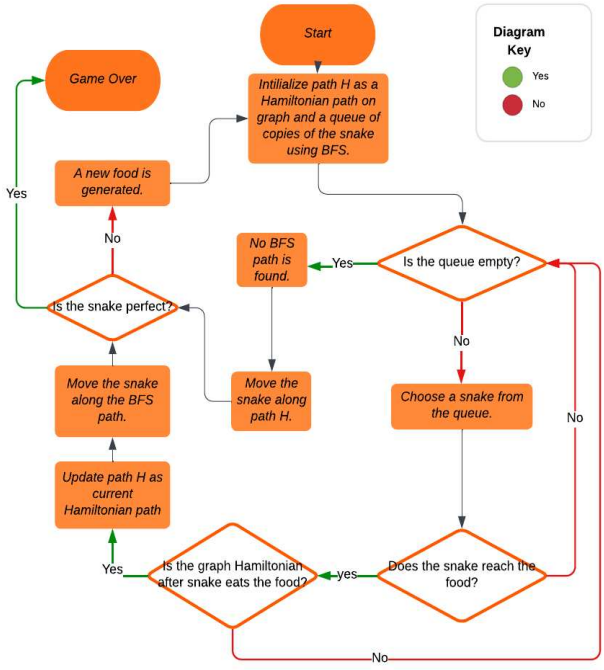
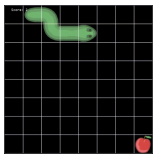
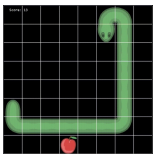


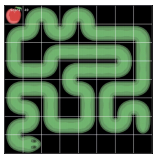
Figure 3: Enhanced Hamiltonian Circuit Algorithm



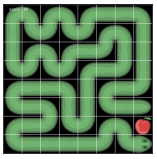
(a) Initial Snake



(b) 25% Progress



(c) 80% Progress



(d) Final Perfect Snake

Figure 4: Perfect Snake on a 8*8 Grid

REFERENCES

[1] Md. Rafat Rahman Tushar and Shahnewaz Siddique. 2022. A Memory Efficient Deep Reinforcement Learning Approach For Snake Game Autonomous Agents. In *2022 IEEE 16th International Conference on Application of Information and Communication Technologies (AICT)*. 1–6.

A Novel Scaffolded Assessment Bridging Concepts and Code

Nathaniel Weinman
nweinman@berkeley.edu
University of California, Berkeley
Berkeley, CA, USA

Jack Boreczky
jack.boreczky@berkeley.edu
University of California, Berkeley
Berkeley, CA, USA

Armando Fox
fox@berkeley.edu
University of California, Berkeley
Berkeley, CA, USA

ABSTRACT

We propose a new exercise format for an upper-division software engineering course. The exercises scaffold the gap between the "box and arrows" representation of a complex software architecture and the nuts-and-bolts realization of that architecture in code. A qualitative study with 12 upper-division computer science students explored the effectiveness of these exercises as a teaching tool and opportunities to improve them. All participants desire for these new exercises to be incorporated into their course, self-rating the exercises as educational and confidence-boosting, suggesting these new types of exercises will pave the way for more confident and competent software engineers.

CCS CONCEPTS

• **Applied computing** → **Interactive learning environments**; • **Software and its engineering** → *Patterns*; *Organizing principles for web applications.*

KEYWORDS

Computing Education, Upper-Division, Web Architecture, Parsons Problems

ACM Reference Format:
Nathaniel Weinman, Jack Boreczky, and Armando Fox. 2024. A Novel Scaffolded Assessment Bridging Concepts and Code. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635509

1 MOTIVATION

A key skill in working with complex software systems is problem decomposition—the ability to decompose a complex goal or workflow into smaller, more manageable goals, with appropriate boundaries and with an understanding of the division of responsibilities (control flow and data management) among the subtasks. Software engineering textbooks contain no shortage of such "box and arrows" representations of subsystems or subgoals, but students must bridge the gap between these abstract representations of patterns and the realization of those patterns in code: what subsystems are involved in realizing the pattern? What data is produced or consumed by each? How is control flow managed among them? We prototype and perform an initial qualitative evaluation of two interactive exercises that scaffold both subgoal decomposition and realizing subgoals in code.

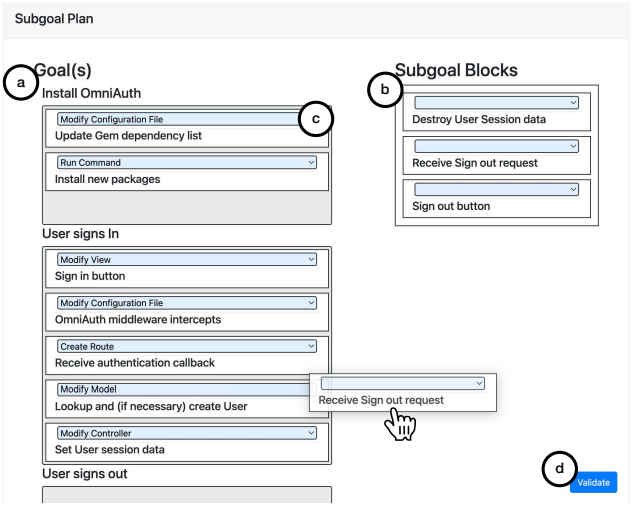


Figure 1: Given a list of goals (a), students drag Subgoal Blocks (b) into the appropriate goal on the left, ordered within each goal by execution flow, and specifying where in the Rails MVC framework the corresponding code should go (c). Students can validate partial answers (d) at any point.

2 RELATED WORK

Past work substantiates the benefits of problem decomposition or top-down thinking [1, 2, 8], yet many students struggle to use this skill effectively [4, 5], resulting in several efforts to create tools to scaffold the process [6, 7]. Closest to our work is probably [3], in which participants progress from selecting and ordering plain-language plan goals to filling in small code snippets; they find that participants from a secondary data-oriented programming course could complete scaffolded code writing and debugging activities after only 30 minutes of instruction. Our work similarly introduces an explicit layer between the high-level goal of the problem statement and the low-level implementation details of code.

3 EXERCISES

The problem we chose involves decomposing the workflow used when third-party authentication occurs in a Web application. This is a complex workflow with three participating entities that "touches" many parts of the application. Instructors had noted that students struggled with understanding this workflow and implementing it in their own applications, and had created a draft code-writing

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635509

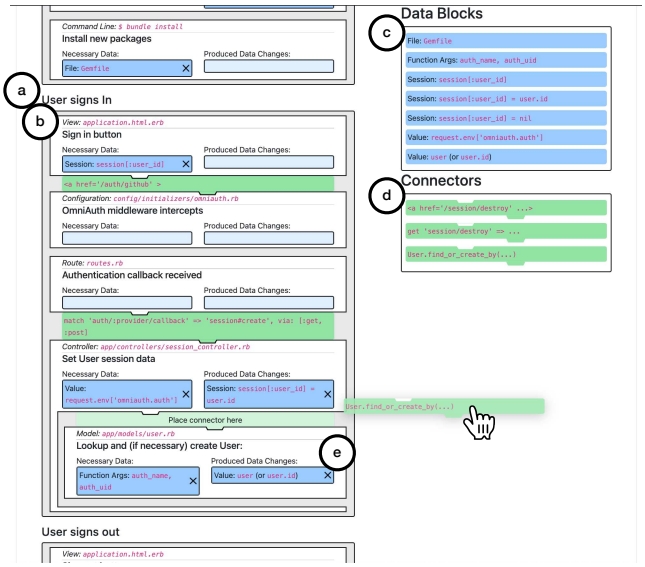


Figure 2: Students are given the same list of goals and subgoals (a) as the previous exercise, but each now specifies (b) the file where its code belongs. Students drop Data Blocks (c) to specify necessary data and produced data changes by each subgoal. Students must also drag Connectors (d) between subgoals where appropriate. Students can remove incorrectly placed blocks (e) by pressing "X".

assignment covering this topic by asking students to add third party authentication to their application using an open source library.

Figure 1 shows the first exercise, in which students arrange and order subgoals decomposing this workflow. Additionally, students must specify where in the Rails MVC framework they would modify code to implement the subgoal. Figure 2 shows the second exercise, in which students indicate the specific implementation of each subgoal, both by placing connectors (code snippets indicating control flow) between subgoals and by specifying what data is *consumed* or *produced* by each subgoal, such as function arguments, variable names, file names, and so on. When students request feedback, the first incorrect component is highlighted in red.

4 PILOT STUDY

We ran a small qualitative study ($N = 12$) to explore the exercises' potential efficacy for learning. Participants were enrolled in the course for which the exercises were designed, and were paid $15 for one hour of time via videoconferencing. After a brief interview and a tutorial on the new exercise type, participants read a problem statement and worked on that same problem through three exercises: (i) create a problem decomposition as complete as possible without any tooling (10 minutes); (ii) solve the problem as a Subgoal Decomposition exercise (Fig. 1, 25 minutes); (iii) solve the problem as a Data Flow exercise (Fig. 2, 25 minutes). After each part, participants were asked three questions to self-assess their confidence in their understanding.

Comments from the interviews were promising: Instead of…"fill in this thing" [P4] code-writing, [the exercises] help the student "actually understand [how] the Model-View-Controller framework" applies to the problem [P5]. The exercises "help reinforce relevant ideas" [P3] to "get the structure" [P11] of the solution, and help "construct knowledge of the language"[P3] by giving them examples of "a canonical way of doing things" [P7].

By not providing access to a code editor, these exercises do not allow a "code first ask questions later" [P4] approach, which sometimes resulted in "sitting there for hours on end trying to figure out" [P5] why code doesn't work, whereas with these exercises, "if [they were] lost, [they could] find out why". Indeed, in the unaided decomposition exercise, 6 of 12 participants would have started writing code within less than 10 minutes of reading the problem, despite not necessarily understanding all the subgoals, and one participant noted that it is "scary that you can finish a [programming assignment] without understanding the structure" [P11]. Participants also highlighted the benefits of being able to focus on the whole problem because this exercise allowed them to keep it all in their head.

5 CONCLUSION

We present and evaluate two new exercises to teach problem decomposition and architectural patterns to upper-division students. These exercises focus on topics that are difficult to learn with traditional code writing exercises, enabling deliberate practice for students to master higher-level patterns and processes.

REFERENCES

[1] Soumya Basu, Albert Wu, Brian Hou, and John DeNero. 2015. Problems before solutions: Automated problem clarification at scale. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*. 205–213.

[2] Andrew Burton-Jones and Peter Meso. 2008. The effects of decomposition quality and multiple forms of information on novices' understanding of a domain from a conceptual model. *Journal of the Association for Information Systems* 9, 12 (2008), 1.

[3] Kathryn Cunningham, Barbara J. Ericson, Rahul Agrawal Bejarano, and Mark Guzdial. 2021. Avoiding the Turing Tarpit: Learning Conversational Programming by Starting from Code's Purpose. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 61, 15 pages. https://doi.org/10.1145/3411764.3445571

[4] Kyungbin Kwon. 2017. Novice programmer's misconception of programming reflected on problem-solving plans. *International Journal of Computer Science Education in Schools* 1, 4 (2017), 14–24.

[5] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A Study of the Difficulties of Novice Programmers. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (Caparica, Portugal) *(ITiCSE '05)*. Association for Computing Machinery, New York, NY, USA, 14–18. https://doi.org/10.1145/1067445.1067453

[6] Alexandra Milliken, Wengran Wang, Veronica Cateté, Sarah Martin, Neeloy Gomes, Yihuan Dong, Rachel Harred, Amy Isvik, Tiffany Barnes, Thomas Price, and Chris Martens. 2021. PlanIT! A New Integrated Tool to Help Novices Design for Open-Ended Projects. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 232–238. https://doi.org/10.1145/3408877.3432552

[7] Sally H. Moritz, Fang Wei, Shahida M. Parvez, and Glenn D. Blank. 2005. From Objects-First to Design-First with Multimedia and Intelligent Tutoring. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (Caparica, Portugal) *(ITiCSE '05)*. Association for Computing Machinery, New York, NY, USA, 99–103. https://doi.org/10.1145/1067445.1067475

[8] Ting Song and Kurt Becker. 2014. Expert vs. novice: Problem decomposition/recomposition in engineering design. In *2014 International Conference on Interactive Collaborative Learning (ICL)*. 181–190. https://doi.org/10.1109/ICL.2014.7017768

Discovering Computer Science: A High School CS Course Co-developed by College & High School Teachers

Pauline White
pwhite@siena.edu
Siena College
Loudonville, NY, USA

Robin Flatland
flatland@siena.edu
Siena College
Loudonville, NY, USA

James Matthews
matthews@siena.edu
Siena College
Loudonville, NY, USA

Jesse Moya
jmoya@siena.edu
Siena College
Loudonville, NY, USA

Kathryn Schiller
kschiller@albany.edu
University at Albany
Albany, NY, USA

ABSTRACT

We present a new high school level computer science (CS) curriculum that is a key component of a comprehensive college/high school partnerships program supporting secondary CS education in New York State. The year-long curriculum offers a broad introduction to CS and is designed using best practices for promoting diversity, equity, and inclusion. It was collaboratively designed and implemented by a team of college CS faculty and high school CS teachers participating in a research-practice partnership. This year (AY2023-2024) the curriculum is being taught in 27 school districts representing high-needs, rural, and suburban communities. Preliminary analysis of student data is included.

ACM Reference Format:
Pauline White, Robin Flatland, James Matthews, Jesse Moya, and Kathryn Schiller. 2024. Discovering Computer Science: A High School CS Course Co-developed by College & High School Teachers. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635507

1 INTRODUCTION

Discovering Computer Science (DCS) was developed as a foundational high school level course for all students that promotes diversity, equity, and inclusion. The year-long course draws from established national curricula and resources such as The Beauty and Joy Of Computing [5], Exploring CS [6], and CS UnPlugged [1]. It extends and adapts these materials to fit the state context. The course covers the history, principles, and transformative applications of computer science and includes a comprehensive introduction to programming. Students start by programming in Netsblox, a friendly graphical language that allows them to express themselves by creating interactive games, animations, and stories, while learning the fundamentals of computer programming. Students continue to develop their programming and problem solving skills using the text-based language Python. Finally, students learn how to design their own websites using HTML, CSS, and JavaScript. Weaved throughout the course are projects and activities on the societal impacts of computing. The course meets all of the CSTA and NYS 9-12 CS Learning Standards [3, 8].

The DCS curriculum was developed and implemented collaboratively by college faculty and high school teachers as an important component of a multi-faceted National Science Foundation grant [1] project. The project was designed to increase the number and diversity of students taking high school CS courses by building CS teacher capacity through comprehensive college/high school partnerships. Most of the new CS teachers in the program had little or no formal CS background and were certified in other areas. The DCS course has been a good first course for new CS teachers to offer. They gain confidence and then often move to teaching more advanced dual-enrollment CS courses.

The college faculty developed the first iteration of the DCS course in AY2019-2020 with regular feedback from teachers' classroom experiences. When the COVID pandemic resulted in schools going remote in the Spring of 2020, a core group of three high school CS teachers, two undergraduate CS/Education majors, and a college CS faculty adapted the curriculum for the remote learning that took place in the AY2020-2021 school year. The adapted curriculum was good for remote learning and continues to be effective in-person as well. The most recent round of curriculum improvements has focused on alignment with the new NYS CS Learning Standards [8].

The DCS course is supported by a summer and school year equity focused professional development (PD) program and a vibrant professional learning community. In its first year (AY2019-2020), DCS was offered in six school districts. This year (AY2023-2024) it is being taught in 27 districts; 12 are rural and 11 are high-needs.

2 FIVE VALUABLE COMPONENTS

Here we describe five components of the DCS curriculum that are fundamental to its goals of being a course for all students that promotes diversity, equity, and inclusion and develops a sense of belonging and confidence in the CS classroom.

Spiral Curriculum: Three of the four main units of DCS focus on the development of a strong foundation in computer programming. These units are block-based programming, Python programming,

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635507

[1]This material is based upon work supported by the National Science Foundation under Grant No. 1923378.

and web programming. In each unit, students study the same foundational programming concepts: statements, variables, math and logic, conditions, loops, and functions. Each unit is broken into activities and laboratory exercises that include increasing levels of difficulty and leverage previous knowledge and learning. Each unit includes at least one open–ended culminating project that invites students to showcase what they have learned. The fourth unit of the course focuses on the impacts of computing, starting with the student's individual voice and experience and growing toward their community and the larger society.

Reflective Learning: Throughout the course, students practice reflective learning. Each laboratory and project ends with an entry into the student's Google Site portfolio. Students use their portfolio to highlight key aspects of their work and to reflect on their learning by recording responses to guided questions. Teachers use the portfolios to gain insight into student learning and build a relationship with the student that reaches beyond the curriculum. Together with the students, teachers share particular entries with parents, prospective students, guidance counselors, and administrators.

Diversity, Equity, Inclusion: Teachers have students in the same classroom that differ by grade level, prior computing experience, gender, ethnicity, cultural identities, and learning strengths and challenges. To this end, teachers are supported with ongoing professional development that centers on the DCS Core Values, which include developing a growth mindset, employing equity-based and culturally responsive teaching practices, and utilizing inquiry-based learning techniques The curriculum includes activities for students that affirm their values, celebrate their cultural identities, and invite collaboration and personal expression. Students also participate in a week-long project where they identify issues related to diversity and see others like them having success in the field.

Differentiated Instruction: The DCS curriculum was developed using Universal Design for Learning [2] strategies to provide the best opportunity for every student to have a high quality experience in the course that challenges them appropriately and helps them learn and grow. Examples of strategies included in the curriculum are the use of multiple media, scaffolded labs and activities that include "bells and whistles" activities for more advanced students, and fostering open communication in a collaborative learning environment. Teachers learn and develop effective strategies for productive use of pair programming, fostering a safe and supportive learning environment, and customizing lessons to meet student needs.

Stakeholder Involvement: We provide teachers with guidance and resources around student recruitment and equity in CS via the CAPE model [4]. However, we recognize that teachers alone cannot realize the goal of providing an equitable and inclusive high-quality CS education for every student. That is why we have begun working to involve school guidance counselors and administrators in developing an effective CS education culture in their district. We are also facilitating the exchange of ideas among districts through multi-district meetings and conference presentations.

3 DCS STUDENT DATA

Preliminary analysis of 304 matched pre/post DCS student surveys between 2019 and 2022 showed that student perceptions of their CS skills increased over the course of the school year with high statistical significance ($p < .001$). Student indicators of a sense of belonging in the CS classroom (e.g. felt valued and welcomed) on post-surveys were also high. Table 1 shows example statements and mean student responses. Students rated their level of agreement with the statements on a 1-5 Likert scale where 1 = "strongly disagree" and 5 = "strongly agree", with 3 = "neither agree or disagree". However, student interest in pursuing CS in the future had little to no change from the pre- to post-surveys. Interviews conducted with female DCS students suggested that having a caring female teacher helped female students feel a greater sense of belonging in the classroom where they were a minority. One female student said, "[my teacher] she's so understanding and helpful... And I think that that is one of the reasons why I'm continuing it." The student also appreciated how the DCS course helped her better understand the role of a computer scientist, but she wanted to know much more about daily life in the field before making any future career decisions. While there is a need for additional research on factors influencing students' interest in careers in CS, the data from this study show that quality experiences in one introductory CS course did not lead to statistically significant increases in students' interest in pursuing additional CS opportunities, a challenge also noted in other studies [7, 9]. However, students did develop confidence in their computer science skills and were likely able to make more informed decisions about their future pathways in computer science.

Table 1: Example post-survey results.

Statement	Mean
The teacher for this course made me feel welcomed in this class.	4.39
The teacher for this course encouraged my participation in the class.	4.30
I felt my opinions were valued in this class.	3.84
Overall, I was satisfied with my experience in this class.	4.05

REFERENCES

[1] Bell, T., Witten, I., and Fellows, M. CS Unplugged: An enrichment and extension programme for primary-aged students. https://classic.csunplugged.org/books/, 2015.

[2] CAST. Universal design for learning guidelines. Version 2.2. http://udlguidelines.cast.org, 2018.

[3] Computer Science Teachers Association. CSTA K-12 computer science standards. www.csteachers.org/k12standards/, 2017.

[4] Fletcher, C. L., and Warner, J. R. CAPE: A framework for assessing equity throughout the computer science education ecosystem. *Commun. ACM 64*, 2 (2021), 23–25.

[5] Goldenberg, P., Mark, J., Harvey, B., Cuoco, A., and Fries, M. Design principles behind Beauty and Joy of Computing. In *Proc. 51st ACM Tech. Symp. on CS Education* (2020), p. 220–226.

[6] Margolis, J., Goode, J., and Chapman, G. An equity lens for scaling: A critical juncture for exploring computer science. *ACM Inroads 6*, 3 (2015), 58–66.

[7] Ni, L., Bausch, G., Thomas-Cappello, E., Martin, F., and Feliciano, B. Creating apps for community and social good: Learning outcomes of a culturally responsive middle school computer science curriculum. In *Proc. of the 54th ACM Tech. Symp. on CS Education* (2023), p. 73–79.

[8] NYS Education Department. Computer science and digital fluency learning standards. www.nysed.gov/curriculum-instruction/computer-science-and-digital-fluency-learning-standards.

[9] Sax, L., Newhouse, K., Goode, J., Skorodinsky, M., Nakajima, T., and Sendowski, M. Does AP CS Principles broaden participation in computing? An analysis of APCSA and APCSP participants. In *Proc. 51st ACM Tech. Symp. on CS Education* (2020), p. 542–548.

Guided Undergraduate Training for Shark Segmentation (GUTSS)

Amy Wu*
amy.wu@ufl.edu
University of Florida
Gainesville, Florida, USA

Morgan Cobb*
morgan.cobb@ufl.edu
University of Florida
Gainesville, Florida, USA

Victor Perez
vjperez@smcm.edu
Department of Environmental Studies
St. Mary's College of Maryland
St. Mary's City, Maryland, USA

Christan Grant
christan@ufl.edu
Department of Computer and Information Science and Engineering
University of Florida
Gainesville, Florida, USA

Jeremy A. Magruder Waisome
jam323@ufl.edu
Department of Engineering Education
University of Florida
Gainesville, Florida, USA

ABSTRACT

As artificial intelligence (AI) becomes more ubiquitously used in education, the need to provide instruction around AI skills will increase. This work presents an opportunity for students to develop image manipulation skills through segmentation. Guided Undergraduate Training for Shark Segmentation (GUTSS) is a mobile application that enables students to use these skills while simultaneously learning about marine anatomy. To make the connection between AI and science education, in-service science teachers can use the software inside and outside of the classroom to help students learn about different aspects of shark anatomy through the GUTSS. Through the application, teachers can enrich their classroom curriculum with technology, share materials, grade assignments, and view their students' work. GUTSS uses open-set object detection, image segmentation, and image manipulation to assist users with organ identification. Gamification within the application will make learning shark anatomy more engaging to students. Prior work from the NSF ITEST Award indicates teachers' willingness to integrate AI concepts into the classroom aligned with state standards. Further wireframing of the application from instructor perspectives is ongoing, to incorporate teacher viewpoints from mixed-methods survey responses on AI usage and anatomy. Future work aims to supplement shark dissection images from AI generation of anatomical image data sets, by collecting these data from students and teachers. Ultimately, the application will be able to identify non-aquatic organisms' anatomical features as a tool for students to learn.

CCS CONCEPTS

• **Software and its engineering** → Designing software; • **Applied computing** → *Interactive learning environments*; • **Human-centered computing** → *Scientific visualization*; • **Computing methodologies** → **Image segmentation**.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635573

KEYWORDS

AI, ML, Segmentation, Anatomy, Education, Science, Gamification

ACM Reference Format:
Amy Wu, Morgan Cobb, Victor Perez, Christan Grant, and Jeremy A. Magruder Waisome. 2024. Guided Undergraduate Training for Shark Segmentation (GUTSS). In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635573

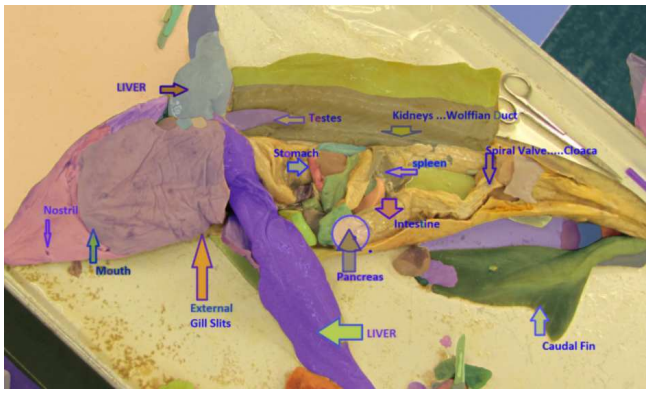


Figure 1: GUTSS application segments a dogfish dissection image. It labels many of the internal and external structures to aid K-12 students. (This is a modified image [5].)

1 INTRODUCTION

Teaching the younger generation about AI needs to start with education's fundamental base: the teachers. Despite a lack of in-service teachers on AI integration with their existing curriculum, prior research indicates teachers' willingness in professional development opportunities for advanced technological concepts, like AI [8]. During a week-long teacher professional development workshop, middle school science teachers learned how to introduce AI concepts and machine learning models using fossil shark teeth in their classrooms throughout an academic year. Teacher and student data collected from the project indicated increased self-efficacy for

both students and teachers around understanding and using AI [8]. These preliminary data led authors to consider other ways science teachers might use AI with sharks and similar species.

Many K-12 schools cannot access the necessary equipment required for hands-on animal dissections for students. The experience requires not only a financial cost for an operation with a limited number of student participants but also a controlled environment to maintain order and minimize clutter around the cadaver. After the anatomy lesson, students cannot revisit the activity at their own pace, rather only relying on their recollection.

We aim to provide an accessible and affordable anatomy learning alternative to secondary school students and teachers by using deep learning and educational gamification. In particular, GUTSS provide affordability through its open-sourced content. Users can use and adapt to how its interface would be used, catering to how the students understand marine biology in the classroom at their own pace. Related works like the MERLIN Project and Seek by iNaturalist use AI to reinforce content comprehension through interactions [4]. Therefore, we propose *GUTSS*, a mobile application to simulate shark anatomy dissection with AI in the classroom. Teachers will be able to create their own organ identification assignments for their students to earn points. GUTSS uses zero-shot object detection, image segmentation, and image manipulation, to automatically label anatomical structures to evaluate students and provide students feedback on their learning progression.

2 THE GUTSS APPLICATION

The GUTSS application has a teacher-mode and a student-mode. Teacher-mode allows the invitation of students to complete a set of assignments. Teachers may add new or pre-labeled images as assignments. Students in student-mode are assigned a classroom and activities to complete. Users can use either mode to take pictures and initiate an auto segmentation. In student-mode, students are expected to label segmented portions for reward points, demonstrating how much the student achieved thus far. Instructors' survey responses will strengthen GUTSS development for classroom usage for teacher-mode and student-mode.

GUTSS employs a fine-tuned Segment-Anything-Model (SAM) [2] and Grounding DINO [3] to identify organs and other animal anatomical structures (See Figure 1). Specifically, GUTSS employs zero-shot object detection, image segmentation, and image manipulation to grade the student assignments for points as part of an educational game. First, the SAM model segments images of organs; GUTSS will then highlight the shark organ through semantic segmentation [2]. Afterwards, the student will answer the identified organs' labels for the assignment. GUTSS will grade the student labels with Grounding DINO. [3].

The app in initial testing will use small anatomical data-sets, leaving room for efficiency improvement [7]. The teachers' added labels and corrections to student submissions are added to GUTSS' internal training data set. As assignments are submitted, GUTSS is retrained and improves in accuracy.

With gamification, students increase their class material comprehension through the GUTSS mobile application's gameplay [1]. AI enrichment and reward system makes the online learning process more engaging for the students [4]. AI will also familiarize with the user, offering dynamic anatomical assessments on course content in response to his or her need of shark understanding.

We received nine responses to a questionnaire on K-12 instructors experience of technology platforms, AI knowledge, and shark dissections. The results are that majority use AI and are willing to use it in education, seeing it as an potential alternative perhaps toward animal dissection support. Moreover, we are looking to send out a follow-up survey to said instructors on potential key features and user interaction with course material and AI usage. Upon further revisal and review, the application will be sent out to be tested live in classrooms with shark dissection.

As a follow-up study topic, shark dissection is a precursor to include other organisms to increase our understanding on biodiversity. Therefore, GUTSS will meet the increasing student demand to understand the way AI operates [6]. It encourages them to think about AI impact on shark dissection and elsewhere.

3 SUMMARY

Instructors are fundamental to students' learning. Their eagerness to network led to conferences like shark teeth identification with AI. As teachers learn how AI works, students learn it from the teachers.

One concept involves diving into shark anatomy. However, secondary school classrooms lack the proper equipment and secure environment. As a result, many students are unable to dissect sharks. Therefore, we aim to develop GUTSS, an accessible and affordable mobile application with AI and gamification. It provides incentive for the students' achievements in game. Furthermore, it will offer more engagement in classroom learning, gaining more accuracy automatically through students' labels and instructor feedback. Hence, schools will not only have a humane alternative to dissecting sharks, but also at a lesser cost.

REFERENCES

[1] Mónica Stambuk Castellano, Ignacio Contreras-McKay, Andrés Neyem, Emilio Farfán, Oscar Inzunza, Nicolás E Ottone, Mariano Del Sol, Carlos Alario-Hoyos, Macarena Soto Alvarado, and R Shane Tubbs. 2023. Empowering human anatomy education through gamification and artificial intelligence: An innovative approach to knowledge appropriation. *Clinical Anatomy* (2023).

[2] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. 2023. Segment Anything. *arXiv:2304.02643* (2023).

[3] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. 2023. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499* (2023).

[4] Mai Neo, Chin Poo Lee, Heidi Yeen Ju Tan, Tse Kian Neo, Yong Xuan Tan, Nazi Mahendru, and Zahra Ismat. 2022. Enhancing Students' Online Learning Experiences with Artificial Intelligence (AI): The MERLIN Project. *International Journal of Technology* 13, 5 (2022), 1023.

[5] RetLaw. 2011. *DogFish Shark Dissection.* Bio 324 Vert Zoo CSULB. http://bio324vertzoo.blogspot.com/2011/02/dog-shark-dissection.html

[6] David Touretzky, Christina Gardner-McCune, Fred Martin, and Deborah Seehorn. 2019. Envisioning AI for K-12: What should every child know about AI?. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 9795–9799.

[7] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. 2018. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 8769–8778.

[8] Jeremy A Magruder Waisome, Dennis R Parnell, Pasha Antonenko, Brian Abramowitz, and Victor Perez. 2023. Board 385: Shark AI: Teaching Middle School Students AI Fundamentals Using Fossil Shark Teeth. In *2023 ASEE Annual Conference & Exposition.*

Interactive Learning Modules for Fostering Secure Coding Proficiency in Introductory Programming Courses

Guangming Xing
guangming.xing@wku.edu
Western Kentucky University
Bowling Green, KY, USA

Gongbo Liang
gongbo.liang@tamusa.edu
Texas A&M University – San Antonio
San Antonio, TX, USA

Tawfiq Salem
tsalem@purdue.edu
Purdue University
West Lafayette, IN, USA

ABSTRACT

In this poster, we introduce a set of modules designed to enhance security education in introductory programming courses. The modules cover a range of prevalent security concerns, including topics such as integer overflow, buffer overflow, and user input validation. These critical security problems are addressed to equip students with essential knowledge and skills in secure coding. The modules feature code examples within sandbox environments, as well as presented using the Python Tutor code visualizer. The eight modules align seamlessly with chapters found in popular introductory programming courses. Each example has been implemented in C++, Java, and Python, ensuring broad applicability across different programming languages.

ACM Reference Format:
Guangming Xing, Gongbo Liang, and Tawfiq Salem. 2024. Interactive Learning Modules for Fostering Secure Coding Proficiency in Introductory Programming Courses. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635528

1 INTRODUCTION

Over the last two decades, cybersecurity has increasingly found itself at the forefront of computer science and media attention. This observation is amplified by the fact that the current ABET CAC curriculum lists secure computing as part of the curriculum requirements. Although numerous colleges and universities now provide security courses within their computer science programs, most universities do not mandate these courses for students pursuing an undergraduate computer science degree. Consequently, many students graduate without fundamental security training and the knowledge of secure coding practices [3]. Integrating security education into lower divisional courses, particularly in CS 0, 1, and 2, has been a slow process. Instructors in introductory programming courses often find themselves lacking motivating examples and support materials to cover security topics. In this paper, we present eight modules that have interactive learning materials that can be readily used in computer science 0 or 1 courses. With the help of live sandboxes and code visualizers, students can get hands-on experiences in important security concepts such as buffer overflow and integer overflow with clarity and ease. Our interactive and engaging user interface makes security education more accessible to students in introductory courses.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635528

2 LIST OF MODULES

Several papers have studied the importance of incorporating security education in introductory programming courses and have developed very helpful content that can be easily adapted [2, 5]. We have developed a list of modules that cover similar content. Below is a list of topics and the student learning outcomes for each module.

(1) **Integer Overflow:** Students will comprehend the risks associated with numeric variables surpassing their maximum or minimum values, leading to potential vulnerabilities.
(2) **User Input Validation:** The significance of user input validation is emphasized, along with an informal discussion of SQL injection.
(3) **Variable Naming:** The impact of variable naming on a program's logic and its potential to evolve into a security concern is covered. The module underscores the role of meaningful variable names in promoting both code clarity and security.
(4) **Scope of Variables:** In-depth coverage of proper variable scope management. The module discusses how varying scope rules can alter program behavior.
(5) **Basic Cryptography and Substitution Cipher:** Basic substitution ciphers are discussed.
(6) **Buffer Overflow/index out-of-bounds:** The intricacies of buffer overflows and array index out-of-bounds exceptions are explored, underscoring their potential to lead to critical security breaches. Mitigation strategies, including array bounds checking and secure coding practices, are covered.
(7) **Encapsulation and Data Hiding:** Covers how these principles preserve data integrity and control access, emphasizing techniques such as access modifiers, getters, setters, and encapsulation as a security measure.

The modules can be covered as an additional section after a chapter in a traditional textbook. Drawing on the structure of Daniel Liang's textbook [4], for instance, the integer overflow can be covered after Chapter 2 (Elementary Programming), while user input validation can be covered after Chapter 3 (Selections).

3 CONTENT DELIVERY METHODOLOGY

Even though the list of topics presented in the above section is similar to several existing tools, our methodology for presenting the materials differs. We employ a three-pronged approach, emphasizing a combination of educational content, hands-on practice,

and robust assessment. To illustrate each concept, we utilize two helpful tools: a sandbox and code visualization. The sandbox provides students with an interactive environment to make real-time changes, enabling them to witness the immediate effects. Code visualization using the Python Tutor [1] visualizer gives students a meticulous, step-by-step observation of program execution. These practical experiences are complemented by comprehensive assessments designed to evaluate students' understanding and application of the presented material. The integration of assessments ensures that students not only grasp the theoretical foundations but also demonstrate their ability to implement secure coding principles effectively.

In introductory programming courses, sandboxes provide students with a dynamic and hands-on learning experience. These virtual environments offer a secure and controlled space where students can actively engage with code examples and experiment with secure coding practices. By allowing students to manipulate, run, and test code within the sandbox, they not only grasp the theoretical foundations but also develop the critical skill of applying their knowledge to real-world scenarios. Figure 1 below demonstrates a sample sandbox, showcasing issues related to integer overflow.

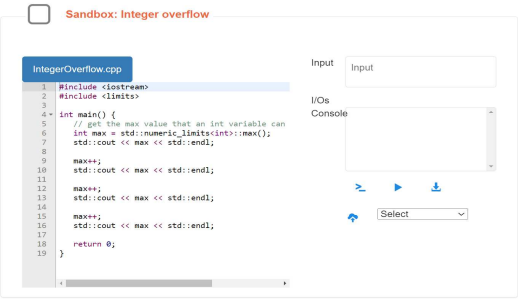


Figure 1: Sandbox for Integer Overflow

Using the code visualizer offers several distinct advantages in the field of programming education. This interactive tool provides students with a visual representation of how their code executes. Each change that may affect the security of a program can be illustrated by the change of the variables in memory. Additionally, the Python Tutor code visualizer supports multiple programming languages, making it a versatile and adaptable resource for various courses. Figure 2 provides a sample code visualization, illustrating the concept of buffer overflow. It allows students to clearly see the effects of the overflow and the resulting consequences.

The third component of the system is assessment. In addition to the conventional multiple-choice questions, we integrate code development questions. These questions empower students to take a hands-on approach, allowing them to assess their comprehension of secure coding principles and put them into practice. An example exercise prompting students to verify the validity of a user password is shown in Figure 3.

4 CONTRIBUTIONS AND FUTURE WORK

We have presented a thoughtful exploration of the modules tailored to bolster the realm of security education in introductory programming courses. By integrating sandbox environments, code

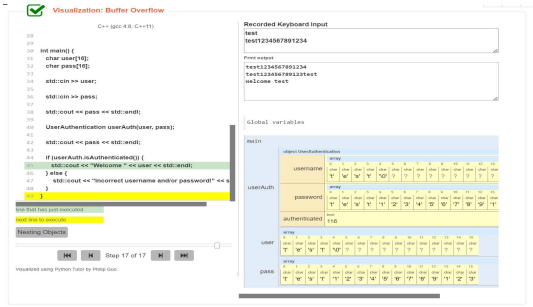


Figure 2: Visualizing Buffer Overflow

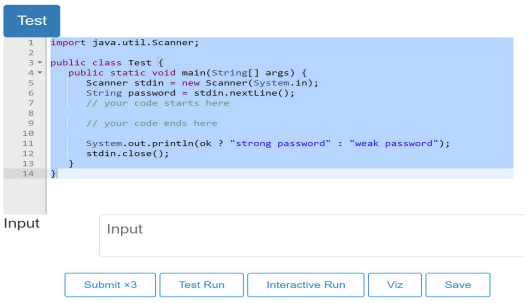


Figure 3: Sample Exercise

visualization tools, and built-in assessment tools, we have provided a dynamic and interactive learning system for security education in introductory programming courses. These pedagogical approaches effectively engage learners, offering them a better understanding of security concepts.

We aim to broaden the topics in the modules by seeking collaborations at the conference to create a comprehensive toolkit.

REFERENCES

[1] Philip J. Guo. 2013. Online Python Tutor: Embeddable Web-Based Program Visualization for Cs Education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 579–584. https://doi.org/10.1145/2445196.2445368

[2] Siddharth Kaza and Blair Taylor. 2018. Introducing Secure Coding in Undergraduate (CS0, CS1, and CS2) and High School (AP Computer Science A) Programming Courses: (Abstract Only). In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 1050. https://doi.org/10.1145/3159450.3162366

[3] Jessica Lam, Elias Fang, Majed Almansoori, Rahul Chatterjee, and Adalbert Gerald Soosai Raj. 2022. Identifying Gaps in the Secure Programming Knowledge and Skills of Students. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1* (Providence, RI, USA) *(SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 703–709. https://doi.org/10.1145/3478431.3499391

[4] Y. Daniel Liang. 2020. *Introduction to Java Programming and Data Structures*. Pearson, Boston.

[5] Sagar Raina, Siddharth Kaza, and Blair Taylor. 2016. Security Injections 2.0: Increasing Ability to Apply Secure Coding Knowledge Using Segmented and Interactive Modules in CS0. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 144–149. https://doi.org/10.1145/2839509.2844609

LEARNDB: A Comprehensive Toolkit for Database Education

Guangming Xing
guangming.xing@wku.edu
Western Kentucky University
Bowling Green, KY, USA

Tawfiq Salem
tsalem@purdue.edu
Purdue University
West Lafayette, IN, USA

Gongbo Liang
gongbo.liang@tamusa.edu
Texas A&M University – San Antonio
San Antonio, TX, USA

ABSTRACT

In this poster, we introduce LEARNDB (*L*earning *E*nvironment and *R*esource *N*etwork for *D*ata*b*ases), a platform designed to meet the unique demands of database education. The platform offers a comprehensive set of tools designed to facilitate the creation and management of tutorials, exercises, quizzes, and laboratory assignments, encompassing topics spanning from database design to SQL (Structured Query Language) proficiency. We have also developed content that encompasses topics spanning from database design to SQL (Structured Query Language) proficiency, providing students with practical skills and knowledge.

ACM Reference Format:
Guangming Xing, Tawfiq Salem, and Gongbo Liang. 2024. LEARNDB: A Comprehensive Toolkit for Database Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635526

1 INTRODUCTION

Unlike introductory programming courses, which often have abundant resources for exercises and assessments, the resources for introductory database courses are comparatively sparse, particularly regarding live coding exercises. This scarcity of available content poses significant challenges in developing assignments, quizzes, and exams that effectively assess students' understanding of database concepts.

This scarcity not only makes the creation of assignments and assessments more time-consuming but also puts additional pressure on instructors to design unique, relevant, and challenging exercises. It can also limit the variety and depth of questions that can be posed to students, potentially hindering their comprehensive learning experience. Addressing this resource gap is essential to facilitate effective teaching and evaluation in database courses.

The main purposes of LEARNDB are to support SQL and database design, two pivotal topics in the realm of database education. SQL, as the language for managing and querying databases, is the fundamental skill required for anyone entering the field of data management. On the other hand, database design is the cornerstone of building robust, efficient, and scalable schemas for information systems. It involves understanding how to structure data tables, establish relationships, and normalize databases to minimize redundancy and ensure data accuracy. By focusing on these two key areas, our system provides learners with a solid foundation in database management, equipping them with the essential skills needed for both practical application and theoretical understanding in the ever-evolving world of data management.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635526

2 LITERATURE SURVEY AND EXISTING WORKS

Numerous educational systems, spanning both commercial and open-source domains, have emerged to support database education. One notable instance is Gradiance [1], a platform developed by Jeffrey Ullman, which offers a comprehensive array of practice problems covering database design, relational algebra, and related topics. In SQL education, several tools and platforms, such as SQLFiddle [2], SQLZoo [3], and zyBooks [5], provide valuable resources and interactive learning experiences. SQLFiddle stands out for its versatility, supporting multiple database systems and offering real-time query execution, though it lacks structured lessons. SQLZoo excels in providing a structured curriculum for beginners, but it may not delve deeply into advanced SQL topics. zyBooks provides a comprehensive course for database education, but there is limited customization for database courses.

3 KEY SYSTEM FUNCTIONS AND CAPABILITIES

LEARNDB provides an extensive array of functionalities designed to cater to the specific requirements of database education. The subsequent diagram delineates the primary functions available to both instructors and students.

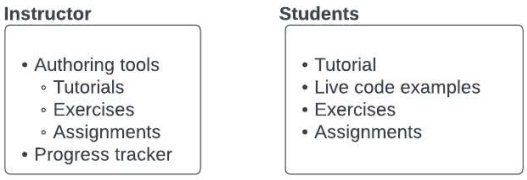


Figure 1: Key Functions

3.1 Functions for Instructors

Firstly, our authoring functions allow instructors to create a dynamic and engaging learning environment when needed. Instructors can effortlessly construct a sample database, and provide code examples, ensuring that students have access to real-world scenarios and practical applications of database concepts. This not only enhances comprehension but also bridges the gap between theory and practice, a critical aspect of effective database education.

Secondly, our platform's intuitive interface equips instructors with versatile tools to craft exercise questions. From multiple-choice

questions that assess theoretical knowledge to matching exercises that challenge students' ability to make connections, and from fill-in-the-blank questions that test code comprehension to coding questions that encourage hands-on problem-solving, LEARNDB covers the entire spectrum of assessment techniques. Leveraging OpenAI's AI-powered data generation, creating a fresh dataset for SQL coding exercises can be accomplished in a few minutes. The system additionally enables instructors to replicate an existing question, facilitating minor revisions.

Lastly, our platform provides seamless administration of quizzes and assignments specific to database courses. Instructors can effortlessly create, customize, and schedule assessments, ensuring they align with the course curriculum and learning objectives. Students can then access these assessments at their convenience, and instructors can track their progress and performance through detailed analytics and reporting tools. This feature streamlines the assessment process, freeing up instructors' time to focus on facilitating meaningful classroom interactions and providing timely feedback.

3.2 Functions for Students

The student interface follows a user-friendly and intuitive design, reminiscent of a traditional ebook interface. It allows easy integration of built-in sandboxes, presenting students with a dynamic and interactive learning environment. Students may seamlessly apply the SQL concepts they encounter in their instructional materials directly into practice. Figure 2 shows a sample sandbox activity:

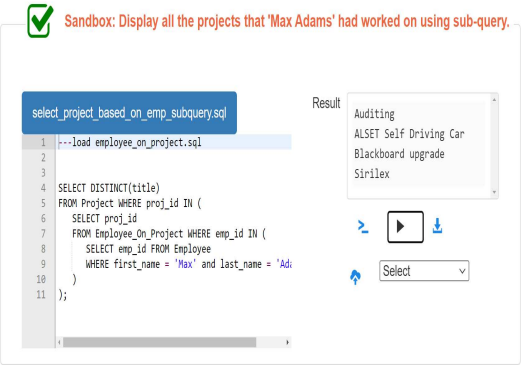


Figure 2: SQL Sandbox

One of the standout features of the student interface is the inclusion of exercises, which serve as pivotal components of the learning experience. These exercises encompass a diverse range of challenges, with a particular emphasis on coding exercises. Figure 3 shows a sample exercise:

The traditional ebook elements, coupled with the inclusion of built-in SQL sandboxes and coding exercises, create a comprehensive and engaging student interface within LEARNDB. This immersive experience empowers students to bridge the gap between theory and practice, ultimately fostering a deeper and more practical understanding of database systems concepts and SQL.

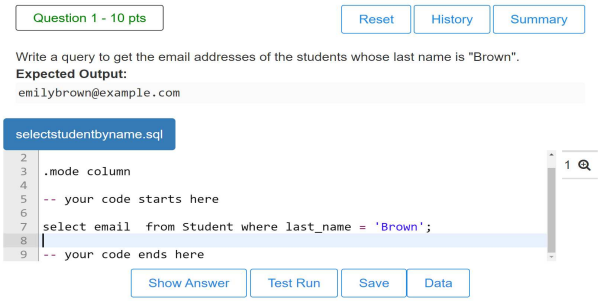


Figure 3: SQL Coding Exercises

3.3 Design of Content

Based on the findings from the study [4], our approach to designing queries for database education has been tailored to address the challenges highlighted in the research. Understanding that the process of learning SQL involves encountering and resolving errors, our queries are crafted to encompass a wide range of SQL concepts, including expressions, joins, grouping, and ordering, which are known to invite various types of errors during the query writing process. To mitigate persistent errors, we place particular emphasis on providing clear and concise explanations, along with step-by-step guidance, to help students identify and rectify logical errors and complications. By aligning our query design strategy with the paper's insights, we aim to enhance the learning experience, equip students with the tools to overcome common pitfalls, and ultimately facilitate a more effective and comprehensive understanding of SQL.

4 CONTRIBUTIONS AND FUTURE WORK

LEARNDB is an innovative platform that significantly enhances student learning outcomes in database education. By providing a systematic approach to auto-graded exercises, particularly in SQL-related questions, LEARNDB fosters student engagement and reinforces their grasp of fundamental database concepts.

Additionally, LEARNDB streamlines the instructor's role through an easily editable question library and user-friendly authoring interface. This adaptability enables instructors to tailor exercises to evolving pedagogical needs, ultimately enhancing the educational experience.

At the conference, we seek collaborators who share our dedication to enriching database education. Together, we aim to expand the repository of exercises and questions, empowering educators to deliver impactful learning experiences and supporting students in mastering SQL and database concepts. We welcome interested parties to join us in this exciting initiative.

REFERENCES

[1] Gradiance 2022. *Gradiance.* Retrieved October 2, 2023 from https://www.gradiance.com
[2] SQLFiddle 2023. *SQLFiddle.* Retrieved October 2, 2023 from http://sqlfiddle.com/
[3] SQLZoo 2023. *SQLZoo.* Retrieved October 2, 2023 from https://sqlzoo.net/
[4] Toni Taipalus and Piia Perälä. 2019. What to Expect and What to Focus on in SQL Query Teaching. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19).* Association for Computing Machinery, New York, NY, USA, 198–203. https://doi.org/10.1145/3287324.3287359
[5] zybook 2023. *zybook.* Retrieved October 2, 2023 from https://zybook.com/

A Cross-Case Analysis of Experienced Educators in CS Inclusion

Wei Yan
University of Florida
wei1993yan@ufl.edu

Andrew Bennett
University of Florida
bennettandrew@ufl.edu

Alexis Cobo
University of Florida
awolfson@ufl.edu

Maya Israel
University of Florida
misrael@coe.ufl.edu

ABSTRACT

Educators should provide access to all students with inclusive and equitable computer science (CS) and computational thinking (CT) learning outcomes. Yet access to CS and CT is not always available for students with disabilities. This qualitative cross-case study examined the barriers and strengths three exemplar teachers faced, explored the supports and resources they provided, and presented how these teachers defined successful inclusive CS learning outcomes for their students. The data set included analysis of the semi-structured teachers' interviews and teaching materials. The results included seven successful strategies the teachers identified: 1) using physical computing, 2) pair programming, 3)connecting CS and Individual Education Plans (IEP), 4)applying hands-on activities, 5) CT integration, 6) using CS vocabulary, and 7)open-ended pedagogy. Three resources and supports the teacher provided emerged from the data set as follows: 1) accessible instructional materials, 2) projects with multiple entry points, and 3) essential scaffolding supports. Whereas four barriers teachers faced; 1. subject matter, 2) accessible tools, 3) students receiving support, 3) the role of CS in instruction, and 4) the role of time served as additional findings. The results suggested that school-based practitioners, including administrators, can overcome the barriers and promote successful strategies that lead to asset-based CS-inclusion in the classroom.

CCS CONCEPTS

• **Applied computing** → **Education**; • **Human-centered computing** → **Accessibility**.

KEYWORDS

CS Education, Special Education, Inclusion, Accessibility

ACM Reference Format:
Wei Yan, Andrew Bennett, Alexis Cobo, and Maya Israel. 2024. A Cross-Case Analysis of Experienced Educators in CS Inclusion. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635631

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635631

1 INTRODUCTION AND PROBLEMS

Providing all K-12 students equal access to computer science (CS) education has become an emerging topic, especially for students with disabilities[6]. Most of those students who received special education service were taught alongside their peers in general educational settings [2]. Although studies suggested additional supports and appropriate instructions that enabled students with disabilities to be successful in CS education, there is a lack of practical pedagogical approaches to provide inclusive CS education[2, 3, 5]. Our study aimed to uncover how experienced teachers provide inclusive CS education to students with disabilities and what barriers they face during implementation.

2 METHODOLOGY

This study is a cross-case study[4] which investigated how three exemplar practitioners provided inclusive education for their students. Three educators 1 were purposefully selected for a 45-minute Zoom interview. The interview protocol included ten questions and related probes, which were developed to develop a deep understanding of educators' CS-inclusion implementation. Four researchers were involved in the Zoom interview process. One researcher asked interview questions, while the other three researchers observed and took notes based on the teachers' answers. Data analysis included the transcription of the three teachers' interviews, operationalizing codes based on teachers' interviews, and analyzing across three cases. To ensure inter-rater reliability, two researchers coded twenty percent of the data together to develop a norming protocol. They then coded the remaining interviews separately. When finished the coding process, the two researchers discussed and solved the differences together.

Pseudonym	Grade	Class Setting	Roles
Sarah	5	Inclusive Co-taught	Special ed
Emily	6-12	Inclusive Teaching	General and Special ed
Jennifer	K-6	Inclusive Teaching	CS facilitator

Table 1: Teachers Information

3 RESULTS

As Figure 1 showed, there were three successful strategies all teachers used in their schools: physical computing, pair programming, and connecting CS with IEP goals. And all of them agreed with using multiple entry points in CS instruction. However, there were various resources and barriers they encountered in the classroom.

		Sarah	Emily	Jennifer
Successful Strategies	Physical computing	x	x	x
	Pair programming/Small Group Instructions and learning	x	x	x
	Connecting CS with IEP Goals	x	x	x
	Use of hands-on activities	x		x
	Integration into other subject areas		x	
	Use of CS vocabulary			x
	Open-ended Padegogies		x	x
Resources	Lending libraries			x
	Essential Scafolding supports		x	x
	Multiple Entry points	x	x	x
	Accesible Instractional Materials		x	x
Barriers	Unclear role of CS	x	x	
	Students not receiving supports.		x	x
	Limited tools			x
	Time consuming		x	x

Figure 1: Teachers' Strategies, resources, and their Barriers.

3.1 Case 1: Sarah

Sarah is a special education teacher who taught 5th-grade students with autism. Her school district had a designated computer science period for each school. Thus, she utilized this time and let her students join the general classroom to learn the CS curriculum together with non-disabled peers. Physical computing, pair programming, and hands-on activities are three instructions to facilitate CS learning for students with disabilities. Notably, she found her students with autism could perform as a leader during paring programming, compared to their usual shyness in social situations. To better facilitate students' learning process, Sarah provided them with multiple entry points to start their CS-related projects. Sarah also connected IEP goals with their CS learning and emphasized the importance of making students with disabilities accountable. Although Sarah provided different supports and resources to include all students in her classroom, the unclear role of CS was a significant barrier for her students. Sarah believed students needed to know the purpose of their CS learning, which could better engage them and help them learn.

3.2 Case 2: Emily

Emily started her CS teaching career after taking a constructionism [1] course, encouraging her to apply the open-ended pedagogical approaches within CS. She valued the role of open-ended pedagogy and integrated CS into different subjects through open-ended projects. In addition, she used physical computing and pair programming to engage her students. For example, her students with disabilities often competed in programming competitions involving physical computing challenges in their local school district. Emily's efforts in providing support were the secret to these students' successful learning processes. Emily engages her students in accessible learning content with multiple entry points. Although these supports took her a long time to prepare, Emily insisted everyone should have opportunities to access and retain knowledge. Coincidentally, Emily also combined CS and IEP goals. The difference was she used it to connect CS with other subjects, although she was not asked to measure students' success through the IEP plan. For example, Emily aligned a sixth-grade geometry lesson that connected a student's math goals with the class's work with Scratch. Even though Emily shared so many inclusive things with us, the unclear role of CS frustrated her in her teaching process. She suggested the administrators reexamine CS's role in policy and provide teachers with more resources for an inclusive education.

3.3 Case 3: Jennifer

Jennifer had a similar opinion to Sarah and Emily that physical computing and pair programming were two representative strengths for students with disabilities in elementary-level CS learning. She used Bee-bot, Coding creditor, and Code-a-Pillars for physical computing. At the same time, Jennifer checked with families to ensure students could learn with these tools at home. To teach CS better in the classroom, Jennifer provided students with four supports: 1) different entry points for students to learn, 2) various types of assessment for students to represent themselves, 3) essential scaffolding supports, and 4) small group instruction. What surprised us was she also used open-ended projects based on CS standards. She applied it to measuring students' success in CS learning. For example, Jennifer designed an open-ended assignment for fourth graders to complete. Students could choose different ways to demonstrate competency in meeting the CS standards.

4 CONTRIBUTIONS TO FUTURE WORK

This work elucidates successful instructional strategies, resources, and barriers for CS inclusive education. The findings provide K-12 special education teachers with successful teaching strategies, which could help them replicate inclusive instructions in their school settings. In addition, this study indicated that administrators should consider the role of CS in the K-12 environment and provide district-level support for special education teachers. Furthermore, this study suggested future research should investigate how to utilize the resources for inclusive CS education, such as scaffolding or technology tools.

ACKNOWLEDGMENTS

The authors thank the National Science Foundation for support under award 2031233. All opinions reflected in this paper are those of the authors and not necessarily those of the National Science Foundation. We also thank the three teachers who spent time during interviews speaking with the research team and sharing their instructional materials to help us collect data.

REFERENCES

[1] Idit Ed Harel and Seymour Ed Papert. 1991. *Constructionism.* Ablex Publishing.
[2] Maya Israel, Gakyung Jeong, Meg Ray, and Todd Lash. 2020. Teaching elementary computer science through universal design for learning. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education.* 1220–1226.
[3] Zuzanna Lechelt, Yvonne Rogers, Nicola Yuill, Lena Nagl, Grazia Ragone, and Nicolai Marquardt. 2018. Inclusive computing in special needs classrooms: Designing for all. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* 1–12.
[4] Robert E Stake. 2013. *Multiple case study analysis.* Guilford press.
[5] Sarah Wille, Jeanne Century, and Miriam Pike. 2017. Exploratory research to expand opportunities in computer science for students with learning differences. *Computing in Science & Engineering* 19, 3 (2017), 40–50.
[6] Aman Yadav, Maya Israel, Emily Bouck, Alexis Cobo, and John Samuels. 2022. Achieving CSforAll: Preparing Special Education Pre-service Teachers to Bring Computing to Students with Disabilities. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1.* 196–201.

Advancing Social Justice Through Data Science

John Bartucz
College of Education and Human Development
University of Minnesota
Minneapolis, MN, USA
jbartucz@umn.edu

Ana Smaranda Sandu
Computer Science
Wellesley College
Wellesley, MA, USA
smaranda.sandu@wellesley.edu

Laney Strange
Khoury College of Computer Sciences
Northeastern University
Boston, MA, USA
laneys@northeastern.edu

R. N. Uma
Mathematics and Physics
NC Central University
Durham, NC, USA
ruma@nccu.edu

ABSTRACT

Now, more than ever, there is a need for computer scientists and data scientists to be socially responsible about the algorithms they design and the products they build. We all have to conscientiously analyze the impact of our work, particularly on historically marginalized communities. As we look towards creating a more just future, this means we also need to provide our students appropriate training, so that we can all address existing social inequities and prevent further social inequities from creeping into our work. Data science[1] provides an ideal opportunity to train students to identify existing bias and explore ways to address it. This special session equips educators with tools to bring about this training. The presenters will share their experiences in incorporating data science for social justice in their courses, introducing CS through a social justice lens, and interrogating the collection and use of data. Through moderated group discussions, this special session will collate ideas and strategies from the participants of their respective experiences. The presenters will create and share a repository of the collated ideas to enable a broad swath of educators to incorporate data science for social justice in their respective courses to train the future generation.

KEYWORDS: Computers and Society, Ethics, Data Science, Inclusive computing curricula and pedagogy, Other DEIA or BPC.

ACM Reference format:
John Bartucz, Ana Smaranda Sandu, Laney Strange, R. N. Uma. 2024. Advancing Social Justice through Data Science. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20-23, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3633403

1 OVERALL OBJECTIVE OF THE SESSION

Embedding social justice in computer science courses is not new; it has been discussed since 1972 up to the present [1][2]. However, what has changed is the urgency and drive with which educators want to address social inequities and bring about change. George Floyd s murder greatly stirred people s collective conscience, spurring many, particularly in academia, to take action to address social inequities while encouraging their students to do the same [3][4]. Data science emerged as a convenient tool with which this could be achieved not only in computer science and other STEM fields but also in non-STEM disciplines such as criminal justice and art (for example [5][6]).

The proposed special session aims to support educators to challenge their students, regardless of discipline, to think about social inequities and help address them through a data-informed approach. Providing such training to students has the added benefit of producing a data-literate, conscientious community, willing to challenge the status quo.

2 OUTLINE OF THE SESSION

2.1 Project Overview (~30 minutes)

The session will start with a brief introduction of what the session entails (about 6 minutes) followed by each of the four session presenters describing their experiences in incorporating data science for social justice in their courses. These experiences will range from middle school up to college and from STEM to non-STEM disciplines. Each presenter will take about 6 minutes to share their work. John Bartucz will share his experiences with under-represented K-12 students. Ana Smaranda Sandu will present her work on challenging students understanding of how algorithms affect their daily lives. Laney Strange will discuss her work on centering data science for social justice in CS and DS introductory courses R. N. Uma will share her work in incorporating data science for social justice in a Freshman Seminar course taken by all freshmen regardless of major at her university which is a Historically Black College/University (HBCU) [7][8][9].

2.2 Group Discussions (~30 minutes)

The next 30 minutes will be devoted to moderated group discussions. The session participants will be divided into four groups based on their interest (K-12/Bartucz, STEM/Sandu, CS/Strange, non-STEM/Uma). Each group, led by the respective presenter, will discuss and share their experiences and ideas for

[1] Throughout this session, we use data science" broadly to mean collecting and modeling data, and developing and deploying data science techniques to analyze it.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3633403

using data science for social justice in a variety of courses. Each presenter will moderate the discussions in their group and document the discussions.

2.3 Report Out and Next Steps (~15 minutes)

In the last 15 minutes, the presenters will report out on the ideas, strategies, and methodologies that were discussed in their respective groups. Following the session, the presenters will create a repository documenting, by subject category, how data science for social justice has been – and how it could be – used in specific courses across the disciplines. The presenters will then disseminate this information either through github or a website. This resource will be shared with the session participants and the broader community with an invitation to continue adding to this repository.

3 EXPECTATIONS

This special session is intended for those educators who are interested in infusing (or have infused) their courses with social justice issues and would like to get some (or share their) ideas and resources. The participants will participate in group discussions where they will be encouraged to share their ideas and ask questions. This special session will result in curation of resources that will enable educators to incorporate data science for social justice in their courses. The presenters hope this session will also serve as an opportunity for members of the community who are interested in integrating data science in their classes to find each other and create a support network.

4 SUITABILITY FOR A SPECIAL SESSION

The proposed format suits a special session because it blends elements of a panel, where presenters share their experiences, with smaller group discussions, where input is solicited from every participant. We will focus on foregoing a deficit mindset (in ourselves and our students), and finding ways for everyone to contribute and feel challenged. A highlight of this format is the creation of a tangible product in the form of a curated repository of resources that educators across all disciplines, including those who did not attend the session, will benefit from.

ACKNOWLEDGMENTS

R. N. Uma s initial work on *Data Science for Social Justice* was supported by NSF EES (HRD) #1912408. Her current work is partially supported by NSF CNS#2245958.

REFERENCES

[1] Norman R. Nielsen. 1972. Social responsibility and computer education. In Proceedings of the second SIGCSE technical symposium on Education in computer science (SIGCSE 72), Association for Computing Machinery, New York, NY, USA, 90–96. DOI:https://doi.org/10.1145/800155.805011

[2] Rodrigo Ferreira and Moshe Y. Vardi. 2021. Deep Tech Ethics: An Approach to Teaching Social Justice in Computer Science. In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21). Association for Computing Machinery, New York, NY, USA, 1041–1047. https://doi.org/10.1145/3408877.3432449

[3] Denise Forte. 2021. The Impact of George Floyd s Murder: One Year Later. May 25, 2021. https://edtrust.org/the-equity-line/the-impact-of-george-floyds-murder-one-year-later/ (accessed Aug 1, 2023)

[4] Michelle Garcia. 2021. The Monumental Impact of George Floyd s Death on Black America. May 25, 2021. https://www.nbcnews.com/news/nbcblk/monumental-impact-george-floyds-death-black-america-rcna1021 (accessed Aug 1, 2023)

[5] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. Machine Bias: There s Software Used Across the Country to Predict Future Criminals. And It s Biased Against Blacks. ProPublica May 23, 2016. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing (accessed Aug 9, 2023)

[6] Brian Boucher 2016. Is the Art Market Racially Biased? October 11, 2016. https://news.artnet.com/market/racial-bias-art-auction-market-672518 (accessed Aug 9, 2023)

[7] Rebecca Zulli Lowe, Adrienne Smith, R. N. Uma, and Alade Tokuta. 2022. Synthesizing insights and lessons learned to elevate interest and empower HBCU students to engage in data science. *American Educational Research Association* (AERA) Annual Meeting, 2022.Data Science for Social Justice https://sites.google.com/view/dssj

[8] Rebecca Zulli Lowe, Adrienne Smith, R. N. Uma, and Alade Tokuta. 2022. Defining the essential elements in designing equity-focused data science explorations. 2022 North Carolina Association for Research in Education.

[9] Data Science for Social Justice: https://sites.google.com/view/dssj

Computer Science Curricula 2023 (CS2023): The Final Report

Amruth N. Kumar
Ramapo College of New Jersey
Mahwah, New Jersey, USA
amruth@ramapo.edu

Rajendra K. Raj
Rochester Institute of Technology
Rochester, New York, USA
rkr@cs.rit.edu

ABSTRACT

A joint task force of the ACM, IEEE-Computer Society, and AAAI has updated the computer science curricular guidelines last published in 2013. Included in the updated guidelines, referred to as CS2023, is a revised knowledge model and a new framework for building a customized competency model of the computer science curriculum. Acknowledging that one size does not fit all, the guidelines provide flexibility for computer science programs to structure their curriculum around the competency area(s) they wish to target. Given the pervasiveness of computing in everyday life, the guidelines emphasize the importance of society, ethics, and issues of the computing profession throughout the curriculum. The role of mathematics has been expanded in the guidelines, particularly in response to the latest developments in artificial intelligence. The curricular guidelines will be accompanied by articles written by experts on curricular issues such as ethics, computing for the social good, and accessibility. In the special session, the CS2023 curricular guidelines will be presented, and feedback solicited on facilitating their adoption and adaptation. The session is aimed at computer science educators, administrators, and professionals interested in computer science education.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; **Model curricula**.

KEYWORDS

Computer Science Curricula, Curricular Guidelines, Model Curricula, Knowledge Model, Competency Model, Curricular Practices, Computer Science Education

ACM Reference Format:
Amruth N. Kumar and Rajendra K. Raj. 2024. Computer Science Curricula 2023 (CS2023): The Final Report. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3633405

1 OVERALL OBJECTIVE OF THE SESSION

CS2023 is the latest in a long line of computer science curricular guidelines published by the Association for Computing Machinery (ACM) in 1968 [2], 1978 [3], 1991 [8], 2001 [7], 2008 [4], and 2013 [1]. IEEE-Computer Society (IEEE-CS) has been collaborating with ACM on these reports starting in 1991. For CS2023, given the increased role of artificial intelligence in undergraduate computer science, the Association for the Advancement of Artificial Intelligence (AAAI) also joined the effort. Together, the three professional bodies set up a steering committee of 17 members led by two co-chairs. The steering committee in turn recruited international disciplinary experts in diverse areas of computer science to form the CS2023 task force.

The tasks undertaken by the CS2023 task force included:

(1) The *knowledge model* of the curriculum, consisting of 17 knowledge areas was updated. The concept of core topics was streamlined. Suggestions were provided for packaging courses from knowledge areas and packaging curricula as aggregations of courses.
(2) A *competency framework* was introduced to enable adopters to build a competency model of the curriculum, which includes a framework for systematically identifying tasks, a format for competency specification, and an algorithm to use the competency framework to build a customized competency model.
(3) Experts were invited to write articles on *curricular practices* including social issues, professional practices, pedagogical concerns, and programmatic considerations. The goal of the articles is to summarize the state of the art, inform educators, and attempt to advance computer science education practices.

The work of the CS2023 task force was first introduced in a special session at SIGCSE 2022 [5]. A progress report was provided in a special session at SIGCSE 2023 [6]. The final version of the report will be presented during the proposed special session. In addition, the process of adopting and adapting the curricular guidelines will be discussed.

2 OUTLINE OF THE SESSION

Table 1 provides an outline of the planned session activities. In addition to the two co-chairs, several members of the steering committee will be on hand to discuss details of the CS2023 curricular guidelines. During the discussion session, the audience of educators, program directors, department chairs, and others interested in the future of computer science education will be invited to provide suggestions for facilitating the adoption and adaptation of the curricular guidelines.

3 EXPECTATIONS

The audience of the special session is undergraduate computer science educators, program directors, department chairs, and others

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3633405

Table 1: Session Structure

	Description	Duration
1	CS2023 Overview	5 minutes
2	Knowledge Model	15 minutes
3	Competency Framework	15 minutes
4	Curricular Practices	10 minutes
5	Discussion and Q & A	30 minutes

interested in the future of computer science education. The audience will learn about the latest computer science curricular guidelines, how they have changed since 2013, and how to adopt/adapt them to local needs. Google forms posted on the CS2023 website csed.acm.org will be used to collect feedback from the audience.

4 POST-SESSION COMMUNITY ENGAGEMENT

The work of the task force has been documented and the latest version of the curricular guidelines has been posted at:

https://csed.acm.org

At the site, visitors can also find earlier drafts of curricular content, revision reports, worksheets for using the competency framework, and abstracts of curricular practice articles. The site provides additional opportunities for the community to provide feedback and suggestions.

ACKNOWLEDGMENTS

The CS2023 task force was created and supported by the ACM, IEEE - Computer Society, and AAAI. Members of the steering committee, the 17 Knowledge area subcommittees, and the authors of curricular practice articles contributed greatly to the report. In addition, numerous educators across the world reviewed and commented on earlier drafts and filled out surveys and feedback forms, thereby contributing to the CS2023 final report. Partial support for this work was also provided by the National Science Foundation under Award DUE-2231333.

REFERENCES

[1] ACM/IEEE-CS Joint Task Force on Computing Curricula. 2013. *Computer Science Curricula 2013.* Technical Report. ACM Press and IEEE Computer Society Press. https://doi.org/10.1145/2534860 Accessed: November 14, 2017.

[2] William F. Atchison, Samuel D. Conte, John W. Hamblen, Thomas E. Hull, Thomas A. Keenan, William B. Kehl, Edward J. McCluskey, Silvio O. Navarro, Werner C. Rheinboldt, Earl J. Schweppe, William Viavant, and David M. Young. 1968. Curriculum 68: Recommendations for Academic Programs in Computer Science: A Report of the ACM Curriculum Committee on Computer Science. *Commun. ACM* 11, 3 (March 1968), 151–197. https://doi.org/10.1145/362929.362976

[3] Richard H. Austing, Bruce H. Barnes, Della T. Bonnette, Gerald L. Engel, and Gordon Stokes. 1979. Curriculum '78: Recommendations for the Undergraduate Program in Computer Science— a Report of the ACM Curriculum Committee on Computer Science. *Commun. ACM* 22, 3 (March 1979), 147–166. https://doi.org/10.1145/359080.359083

[4] Lillian Cassel, Alan Clements, Gordon Davies, Mark Guzdial, Renée McCauley, Andrew McGettrick, Bob Sloan, Larry Snyder, Paul Tymann, and Bruce W. Weide. 2008. *Computer Science Curriculum 2008: An Interim Revision of CS 2001.* Technical Report. ACM Press, New York, NY, USA.

[5] Amruth N. Kumar and Rajendra K. Raj. 2022. A First Look at the ACM/IEEECS/AAAI Computer Science Curricula (CS202X). In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education.* Association for Computing Machinery, New York, NY, USA, 2 pages. https://doi.org/10.1145/3478432.3499036

[6] Amruth N. Kumar and Rajendra K. Raj. 2023. Computer Science Curricula 2023 (CS2023): Community Engagement by the ACM/IEEE-CS/AAAI Joint Task Force. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2* (Toronto ON, Canada) *(SIGCSE 2023).* Association for Computing Machinery, New York, NY, USA, 1212–1213. https://doi.org/10.1145/3545947.3569591

[7] The Joint Task Force on Computing Curricula. 2001. Computing Curricula 2001. *J. Educ. Resour. Comput.* 1, 3es (Sept. 2001), 1–es. https://doi.org/10.1145/384274.384275

[8] Allen B. Tucker, Robert M. Aiken, Keith Barker, Kim B. Bruce, J. Thomas Cain, Susan E. Conry, Gerald L. Engel, Richard G. Epstein, Doris K. Lidtke, Michael C. Mulder, Jean B. Rogers, Eugene H. Spafford, A. Joe Turner, and Bruce H. Barnes. 1991. *Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force.* Technical Report. ACM Press and IEEE Computer Society Press, New York, NY, USA.

Microteaching: Binary Heaps, Side-Channel Attacks, Equitable Grading, Java Classes, Loops, and 3D Java

Colleen M. Lewis
Computer Science Department
Univ. of Illinois Urbana-Champaign
Urbana, IL, USA
ColleenL@illinois.edu

Cynthia Bailey Lee
Computer Science Department
Stanford University
Stanford, CA, USA
cbl@stanford.edu

Adam Blank
Computing & Mathematical Sciences
California Institute of Technology
Pasadena, CA, USA
blank@caltech.edu

Maria Camarena
Maywood Center for Enriched Studies
Los Angeles Unified School District
Maywood, CA, USA
MariaCamarena@mfala.org

Manuel Hernandez
Mathematics Department
Bell Gardens High School, MUSD
Math for America - Los Angeles
Bell Gardens, CA, USA
Hernandez_Manuel@montebello.k12.ca.us

Frank Vahid
Computer Science & Engineering
University of California, Riverside
Riverside, CA, USA
vahid@cs.ucr.edu

ABSTRACT

This microteaching session is like Nifty Assignments for instruction. Instead of having the presenters just talk about their teaching, they will simulate how they would actually teach something. Covering a range of topics and grade levels, six educators will demo how they would teach a specific topic. To help identify the pedagogical practices that cut across grade bands and topics, the moderator, Colleen Lewis, will describe how their pedagogical practices connect with education research. The goal of the session is to inspire SIGCSE attendees by highlighting innovative instruction by exceptional educators. Attendees can adopt the content and/or pedagogical practices from each microteaching example.

KEYWORDS

pedagogy; pedagogical content knowledge; innovating teaching

ACM Reference Format:
Colleen M. Lewis, Cynthia Bailey Lee, Adam Blank, Maria Camarena, Manuel Hernandez, and Frank Vahid. 2024. Microteaching: Binary Heaps, Side-Channel Attacks, Equitable Grading, Java Classes, Loops, and 3D Java. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3633406

1 INTRODUCTION AND STRUCTURE

SIGCSE is packed with teaching insights and inspiration. However, we get these insights and inspiration from hearing our colleagues talk about their teaching. Why not *watch* them teach? This session does exactly that!

The goal of the session is to model general pedagogical techniques and pedagogical techniques applicable to teaching specific content. Each of six exceptional educators will teach the audience something (7 minutes each). After each teaching demonstration, Colleen Lewis will draw the attention of the audience to particular pedagogical moves that the instruction included (1 minute). The remaining time will be used for attendees to ask follow-up questions to the presenters of their choice. The structure of a special session provides the flexibility to have six, fast-passed demos, brief comments highlighting pedagogical practices, and ample time for small-group discussion with individual presenters.

These educators and educators in the past [4–7] were selected because they are inspiring, engaging, and effective teachers. Our intended audience is CS educators who want practical and inspiring ideas to take back into their classroom. We intend for attendees to benefit by taking content and/or pedagogical moves from this session directly back to their teaching.

2 EXCEPTIONAL EDUCATORS

2.1 Binary Heap Insert - Cynthia Bailey Lee

This teaching demo will show how I use Peer Instruction (PI) and live coding together in class. Peer Instruction questions both build conceptual knowledge of data structures, and directly support live coding by engaging all students in authoring key pieces of code. The first activity uses a classic "how many of these are correct?" PI question structure, which intentionally leaves unstated *which* are correct, as a means of provoking students to initiate discussion of exactly that when they turn to each other. To support Peer Instruction questions, we will use the Poll Everywhere website/app (no install or account is needed).

Cynthia Bailey Lee is a Senior Lecturer in the Computer Science Department at Stanford. She specializes in team-based and question-based course design, and founded peerinstruction4cs.org to support educators in flipping their computer science classrooms using peer instruction.

2.2 Side-Channel Attacks - Adam Blank

In this teaching demo, I will demonstrate how a topic traditionally taught in a security course can help unify and demystify typical topics in an "introduction to systems" course. I will use an active learning approach to engage and involve the audience as well as

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3633406

a small programming exercise to demonstrate the applicability of the topic.

Adam Blank is a Teaching Professor of Computing and Mathematical Sciences at Caltech. Their teaching approach involves developing new technologies and techniques to enhance the learning of computer science students. They experiment with new techniques and technologies involving technology, human computation and collaboration to improve the classroom experience. They are especially interested in broadening participation in computing via their teaching methods.

2.3 Equitable Grading - Maria Camarena

To foster equity, I implement standards-based grading [1, 8]. In this teaching demo, I will show how I introduce and explain my grading system to my students. Most students are unfamiliar with standards-based grading, and helping them understand how the system helps them understand my commitment to their learning and my expectations for them. This demo aims to provide valuable insights into practical strategies and perspectives for promoting a fair and transparent grading process.

Maria Camarena is a CS teacher at Maywood Center for Enriched Studies in Los Angeles Unified School District. Over the last five years, Maria developed a CS pathway in her school that teaches students in grades 6 through 12. Maria serves as a Computer Science Teachers Association (CSTA) Equity Fellow. For her efforts to include students who identify as women, Hispanic, and/or Latinx, Maria received the 2020 Teaching Excellence Award from CSTA and the Infosys Foundation, and the 2022 National Aspirations in Computing Educator Award from the National Center for Women and Information Technology (NCWIT).

2.4 Java Classes: Class Structure and Writing Constructors - Manuel Hernandez

Object-oriented programming depends on the ability to take common and complex ideas and turn them into objects. Halfway through the AP Computer Science A (a Java course), students are formally introduced to the structure of a class. In the lesson, we will look at a basic structure, apply the basic structure to common objects, and get participants to pair code to create classes and constructors.

Manuel Hernandez has been a high school mathematics teacher for 17 years, and a member of the Master Teacher Cohort in Math for America Los Angeles for 8 years. He was involved in starting a Computer Science program at Bell Gardens High School, in Bell Gardens, California in 2019 and has taught AP Computer Science A for 3 years. He is currently teaching Computer Science Discoveries.

2.5 Loops - Frank Vahid

With loops, we can read a list of values and do things like print only the negative values, but how can we do something like printing the list in reverse? We need to store the list, and that's why we need vectors. We'll show how to append values onto a vector, and how to traverse and print the vector's elements, first forward, then in reverse. We'll point out common mistakes, and maybe even make some of those mistakes along the way! With vectors, programs can be much more powerful.

Frank Vahid has been a Professor of Computer Science at the University of California, Riverside, for 30 years, is co-founder and chief learning officer at zyBooks, and is founder and president of CollegeStudentAdvocates.org. He shifted his research focus to CS education just over a decade ago, passionate about helping more wonderful CS majors achieve their degree goals.

2.6 3D Java - Colleen M. Lewis

In this teaching demo, I will illustrate how I use physical objects to help students understand aliasing in Java using 3D representations [3]. In particular, I use a strategy known as Concreteness Fading [2] wherein I begin with a concrete, physical representation and systematically transition to a more abstract representation.

Colleen Lewis is an Associate Professor of CS at the University of Illinois at Urbana-Champaign. Lewis was previously the McGregor-Girand Associate Professor of CS at Harvey Mudd College. She researches equitable and efficient teaching practices and curates CSTeachingTips.org, a NSF-sponsored project for disseminating effective CS teaching practices.

Colleen also serves as the moderator for the session. This role involves connecting the pedagogical practices in all of the teaching demonstrations with education research. This builds on her training; at the University of California, Berkeley, Colleen completed a PhD in science and mathematics education.

ACKNOWLEDGMENTS

CSTeachingTips.org is funded by the NSF (1821136 and 2144249).

REFERENCES

[1] Joe Feldman. 2024. *Grading for equity: what it is, why it matters, and how it can transform schools and classrooms* (second edition ed.). Corwin, Thousand Oaks, California.

[2] Emily R. Fyfe, Nicole M. McNeil, Ji Y. Son, and Robert L. Goldstone. 2014. Concreteness Fading in Mathematics and Science Instruction: a Systematic Review. *Educational Psychology Review* 26, 1 (March 2014), 9–25. https://doi.org/10.1007/s10648-014-9249-3

[3] Colleen M. Lewis. 2021. Physical Java Memory Models: A Notional Machine. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 383–389. https://doi.org/10.1145/3408877.3432477

[4] Colleen M Lewis, Leslie Aaronson, Eric Allatta, Zachary Dodds, Jeffrey Forbes, Kyla McMullen, and Mehran Sahami. 2018. Five Slides About: Abstraction, Arrays, Uncomputability, Networks, Digital Portfolios, and the CS Principles Explore Performance Task. In *SIGCSE Proceedings*. ACM, 269–270.

[5] Colleen M. Lewis, Christine Bassem, Jason M. Grant, Felienne Hermans, Angel Kuo, Art Lopez, and Beth Trushkowsky. 2023. Microteaching: Ad-Hoc Networks, Binary Heaps, Variables in Hedy, Loops, Lists, and Data Storage. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1218–1219. https://doi.org/10.1145/3545947.3569593

[6] Colleen M Lewis, Kathi Fisler, Jenny Hinz, David J Malan, Joshua E Paley, Manuel A Pérez-Quiñones, and Shikha Singh. 2021. Microteaching: Semantics, Definition of a Computer, Running Times, Fractal Trees, Classes as Encapsulation, and P vs NP. In *SIGCSE Proceedings*. ACM, 766–767. https://doi.org/10.1145/3408877.3432582

[7] Colleen M Lewis, Daniel D Garcia, Helen H Hu, Saber Khan, Nigamanth Sridhar, Bryan Twarek, and Chinma Uche. 2019. Microteaching: Recursion, Coding Style, Creative Coding, Inheritance and Polymorphism, Loops, and the Internet. In *SIGCSE Proceedings*. ACM, 962–963.

[8] Peter Liljedahl. 2021. *Building thinking classrooms in mathematics, grades K-12: 14 teaching practices for enhancing learning*. Corwin, Thousand Oaks, California.

Unlocking Excellence in Educational Research: Guidelines for High-Quality Research that Promotes Learning for All

Monica M. McGill
monica@csedresearch.org
CSEdResearch.org
Peoria, Illinois, USA

Sarah Heckman
sarah_heckman@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Michael Liut
michael.liut@utoronto.ca
University of Toronto Mississauga
Mississauga, Ontario, Canada

Ismaila Temitayo Sanusi
ismaila.sanusi@uef.fi
University of Eastern Finland
Joensuu, Finland

Claudia Szabo
claudia.szabo@adelaide.edu.au
University of Adelaide
Adelaide, South Australia, Australia

ABSTRACT

While there are multiple standards bodies that define characteristics of high-quality, there are limited guidelines on conducting equity-enabling research, particularly in the context of high quality and in computing education. As part of an ACM ITiCSE Working Group in 2023, we engaged in a concept analysis and structured literature review to identify high-impact practices for conducting both high-quality and equity-enabling education research. As a result of this work, we produced a set of guidelines across each major phase of research that integrates characteristics of high-quality education research with those that are necessary for producing research that is designed to honor and meet the needs of various subgroups of learners. Special emphasis is given to the role that the researcher plays in shaping the research based upon how the researcher's lived experiences, perspectives, and training influences their work. During this special session, we will review each set of guidelines and engage attendees in reflection and discussion of them and how they can use the guidelines to enhance their education research.

CCS CONCEPTS

• **Social and professional topics** → **Computer education**; **Computer education programs**; **Computer science education**.

KEYWORDS

Computing education, research, high-quality, equity, primary, secondary, post-secondary, guidelines, CS for all

ACM Reference Format:
Monica M. McGill, Sarah Heckman, Michael Liut, Ismaila Temitayo Sanusi, and Claudia Szabo. 2024. Unlocking Excellence in Educational Research: Guidelines for High-Quality Research that Promotes Learning for All. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3633402

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3633402

1 OVERALL OBJECTIVE OF THE SESSION

The United Nations Sustainable Development Goal 4 focuses on the need to "...ensure inclusive and equitable quality education and promote lifelong learning opportunities for all" [1, p. 1]. As noted by National Research Council, equity should be at the "...forefront of any effort to improve the goals, structures, and practices that support learning and educational attainment for all students." [3, p. 277] In computing education, this framing has been adopted across policies, standards, curriculum, and pedagogy, to support the computer science for all initiative [4].

Equity-enabling education research is research that supports the needs for achieving equity in education, including providing

> ...the evidence needed for decision-makers and educators to advocate, support, and deliver computing education in K-12 classrooms that lead to equitable access, participation, and experiences (and therefore outcomes) among all students . Equity-enabling education research, by its very nature and necessity, must meet quality standards to provide meaningful evidence for particular contexts for which studies have been conducted [2].

But what does equity-enabling research that is designed to meet the goal for CS for All entail? As part of a 2023 ITiCSE Working Group [2], our team developed a set of guidelines that blend characteristics of high-quality with equity-based practices. The overall objective of this session is to introduce the guidelines to those who engage in CS education research. The overall goals of the session will be to:

- Engage attendees in discussion and reflection of characteristics of high-quality research
- Engage attendees in discussion and reflection of characteristics of equity-enabling research
- Discuss and reflection upon the interplay between the two and how they support each other
- Share ample resources for conducting high-quality, equity-enabling education research practices focused on learners of interest to the attendees

2 BENEFITS TO PARTICIPANTS

Participants will walk away with a high-level understanding of how to conduct high-quality, equity-enabling research as well as the many nuances that lead to deeper understanding of what contributes to meaningful, well-situated research. Reflective of the goals, participants will walk away with:

- Deeper understanding of the many aspects of the research design phases
- Deeper understanding of how to embrace their own lived experiences and training to strengthen their research
- Ways to identify and mitigate their own lack of understanding of the lived experiences of participants in their studies
- A set of resources for conducting high-quality, equity-enabling education research practices

3 OUTLINE OF THE SESSION

The outline of our session is presented in Table 1.

Table 1: Session outline.

Description	Min.
Welcome and frame the session (McGill)	5
Introduction to the Guidelines (Heckman)	5
Role of Researcher in Research (Szabo)	10
Framing Problems, Research Questions (Liut)	10
Protocol Development (McGill)	10
Methods (Heckman)	10
Engaging with Participants (Sanusi)	10
Reporting (Heckman)	10
Closing (Szabo)	5

The session will be mostly sharing of the guidelines with attendees and where to find additional resources for learning more. We will also have a Google document set up so that attendees can add their questions and comments. We will closely monitor this and answer questions as they are asked.

The session will be presented by the following computing education researchers:

Monica McGill, Institute for Advancing Computing Education (IACE) (formerly CSEdResearch.org), has two decades of experience as a computer science education researcher and is the founder and CEO of the non-profit IACE. She has been investigating ways to build the capacity of high-quality, equity-focused research during the last 15 years.

Sarah Heckman, North Carolina State University, is an Alumni Distinguished Undergraduate Professor with the Department of Computer Science. She serves as the Director of Undergraduate Programs for the department and she has been studying computer science education for more than ten years.

Michael Liut, University of Toronto Mississauga, is a computing education researcher with a focus on adaptive systems, behavioural interventions, and tool development. Michael enjoys spending his time on developing models, tools, and interventions to improve the academic and educational experience for students.

Ismaila Sanusi, University of Eastern Finland, is a K-12 computer science education researcher with specific interest in working with underrepresented, the vulnerable and under-researched groups.

Claudia Szabo, University of Adelaide, is an Associate Professor of Computer Science. She is computing education researcher focused on understanding the use of learning theories and the software development process of nearly graduating students.

4 EXPECTATIONS

This special session will be of interest to researchers (new and experienced) and evaluators who want a better understanding of how to conduct CS education research that is high-quality as well as serves the needs of supporting all learners. This session may also be of interest to other individuals who support research (i.e., funders). Our session is designed so that it will present a broad overview of each phase of research and we will also provide an ample set of resources for conducting future research that meets these goals. We expect attendees to come ready to reflect on their research and be engaged in thoughtful discussion. No laptops will be needed for this session.

5 SUITABILITY FOR A SPECIAL SESSION

This session is designed to function well as a hybrid event. Specifically, the interactive activities are formatted so that in-person and remote attendees can fully participate. One of the presenters will closely monitor the Zoom chat so that remote attendees can fully engage in the discussion as well.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation, Award No. 2122212.

REFERENCES

[1] Francisco del Cerro Velázquez and Ginés Morales Méndez. 2018. Augmented reality and mobile devices: A binominal methodological resource for inclusive education (SDG 4). An example in secondary education. *Sustainability* 10, 10 (2018), 3446.

[2] Monica M. McGill, Sarah Heckman, Christos Chytas, Michael Liut, Vera Kazakova, Ismaila Sanusi, Selina Marianna Shah, and Claudia Szabo. 2023. Conducting High-Quality Equity-Enabling Computing Education Research. In *n Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 2 (ITiCSE 2023), July 8–12, 2023, Turku, Finland. Manuscript under review.*

[3] National Research Council. 2012. *A Framework for K-12 Science Education: Practices, Crosscutting Concepts, and Core Ideas.* National Academies Press. https://books.google.com/books?id=b2L5VShktGIC

[4] Deborah Seehorn and Lissa Clayborn. 2017. CSTA K-12 CS standards for all. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education.* 730–730.

Selecting BPC Activities: Discussion of the Feasibility, Relevance, and Sustainability of 60+ Activities from BPCnet.org

Briana Scurry
Computer Science Department
Univ. of North Carolina at Chapel Hill
Chapel Hill, NC, USA
Briana.n.e.scurry@outlook.com

Colleen M. Lewis
Computer Science Department
Univ. of Illinois Urbana-Champaign
Urbana, IL, USA
ColleenL@illinois.edu

Luther A. Tychonievich
Computer Science Department
Univ. of Illinois Urbana-Champaign
Urbana, IL, USA
LutherT@illinois.edu

ABSTRACT

This session will help attendees interested in broadening participation in computing (BPC) select specific BPC activities to engage in, either individually or as a group or department. Attendees will be provided with a list of more than 60 BPC activities and will participate in a guided discussion to help select those best suited to them. Topics of discussion will be introduced by the session organizers, discussed in small groups, and then summary points will be shared with all attendees. The guided topic prompts will cover (1) how each activity contributes to BPC; (2) the expected feasibility, impact, and relevance of activities and how to prioritize them; (3) methods of sustaining activities for long enough to see results; and (4) next steps, including implementing activities and incorporating them into departmental BPC plans. At the end of the session, each attendee will have selected multiple BPC activities suited to their goals and institution and have developed their ability to analyze BPC activities along several axes and anticipate how best to sustain them.

ACM Reference Format:
Briana Scurry, Colleen M. Lewis, and Luther A. Tychonievich. 2024. Selecting BPC Activities: Discussion of the Feasibility, Relevance, and Sustainability of 60+ Activities from BPCnet.org. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3633404

1 INTRODUCTION

How should people interested in Broadening Participation in Computing (BPC) pick what activities to engage in?

BPC is a multi-faceted work that seeks to increase the accessibility and inclusivity of computing courses and the workforce; remove barriers to and inequities in the influence, sense of belonging, and outcomes of those engaging in computing; and remove representation inequities, particularly as related to race, ethnicity, gender, and disability. The breadth of this work, coupled with the wide variety of contexts in which this work is done and the wide variety of experience, interest, and resources of those engaging in the work, suggest that many kinds of activities are needed to broaden participation in computing. The wide scope of activities makes selecting good activities for an individual educator or a department challenging.

To assist in selecting BPC activities, the facilitators will present attendees with a list of more than 60 BPC activities covering a variety of BPC goals and suited to a variety of departmental actors [1]. A few example activities from it include:

- **Teaching Audit**: Faculty assess the equity of their own teaching using the "The Teaching Practices Inventory" [5] to identify changes that would better support students from groups underrepresented in computing, and then implement those changes.
- **TA Training**: All new teaching assistants are required to complete inclusive teaching training prior to beginning work. Faculty help organize and teach in these trainings.
- **Faculty Review Student Demographics**: Every year at a department meeting, faculty share data on the representation among undergraduate and graduate students from the institution in comparison to peer institutions and regional K-12 schools.

Attendees will participate in multiple rounds of small-group discussion, helping them consider activities from several perspectives. At the end of the session, each attendee will have selected multiple BPC activities suited to their goals and institution and have developed their ability to analyze BPC activities along several axes and anticipate how best to sustain them.

The National Science Foundation (NSF)'s Directorate for Computer and Information Science and Engineering is one of several groups encouraging every CS faculty member to be engaged in meaningful activities that can help achieve BPC [4]. As part of that initiative, more than 100 Departmental BPC Plans have been collected and published on BPCNet.org [2], each with multiple BPC activities. The session will include some references to how the activities can be integrated into BPC plans submitted to the NSF, though that will not be the main focus of this session.

The goals of this session are to:

- Expose attendees to a variety of activities that can help broaden participation in computing in various ways and suited to various situations.
- Engage attendees in rich discussion of the strengths and weaknesses of BPC activities and how to implement them within a given context.
- Have each attendee come away with specific BPC activities they plan to implement.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3633404

- Provide guidance on next steps, including incorporating activities into a Departmental BPC Plan.

2 STRUCTURE/TIMING

This session will be organized in three parts. First, attendees will be exposed to many BPC activities differing in scope, approach, and goals. Second, attendees will evaluate likely feasibility and impacts of BPC activities. Third, attendees will plan activity lifecycles by anticipating and responding to likely failure modes. To conclude, the facilitators will review the learning goals and resources gained in each part of the session and plans for next steps.

3 DISCUSSION POINTS

The discussion will be directed in part by participant interest, but is expected to explore the following aspects of BPC activities.

Relevance: BPC activities should be selected based upon their relevance within the institution. In most cases, BPC activities are designed to respond to a documented or reported problem within an institution. For example, BPC activities may be designed to remove barriers to participation and persistence, improve the community so that all students feel a sense of belonging, address inequality in opportunities or performance, and/or gather information about current BPC problems. All of these would be in response to a specific documented or reported problem. Many institutions find they have generally the same set of problems, and tend to set similar BPC goals, such as those discussed on BPCnet.org [3]. Individuals from a community that may benefit from a BPC activity can often provide guidance on the relevance and relative priority of a BPC activity.

Impact: Not all BPC activities have the same impact even if successfully implemented. Two principal components of impact are the number of people impacted and the level of impact made on each participant. Activities to train educators can impact many more students than activities that work with students directly. Long-term activities tend to have more impact on each participant than short-term or single-event activities. Some activities may have a beneficial impact on those facilitating the activity, perhaps even exceeding the impact on those attending it; for example, undergraduate students may increase their confidence and sense of belonging from leading K-12 outreach efforts even if those outreach efforts are short in duration and unlikely to have a lasting impact on the K-12 students.

Feasibility: The feasibility of activities is determined by having access to the resources needed to run the activity. While some activities have financial needs, most depend on access to the necessary personnel. Some activities need either many people or significant time from a few people to be effective. Some need facilitators with particular expertise or relatable lived experience. Some need partnerships or existing relationships (e.g., with campus partners, non-profits or schools) to successfully recruit participants. Students, staff, and faculty have different strengths, availability, and expected time at the institution. Even the most relevant and impactful activities are unlikely to be successful if their needs do not match the available resources. Feasibility also depends on aligning activities with local institutional and governmental priorities and restrictions.

Sustainability: High impact generally requires sustained effort: it is not common for any single event, no matter how well organized, to have a large lasting beneficial change in broadening participation in computing. Sustained effort is hampered by a lack of resources, but also by the appeal and excitement of beginning a new effort. The team leading an activity also impacts its sustainability: depending too heavily on short-term employees,students, or on a single organizer can make the activity fragile to changes in personnel. To create lasting impact, institutions often seek to engage in a one-time revision of a course or policy instead of introducing a new program that would require ongoing effort to sustain.

Avoiding Negative Outcomes: It is unfortunately the case that even the best-intentioned BPC efforts can have unintended negative impact. Facilitators with insufficient awareness of the culture of the communities they are interacting with can offend or send negative messages. Outreach efforts that fail to correctly align their activities with the abilities and motivations of the participants can result in reduced confidence and interest in computing. Poorly-designed initiatives to support students from a specific community can communicate an unintended message of presumed inferiority. Some activities are more susceptible to these kinds of unintended negative outcomes than others, but they should be considered and actively avoided for all activities.

4 FACILITATORS

All facilitators serve as BPC consultants for BPCnet.org. In this role, they support faculty in writing BPC plans, review Departmental BPC Plans, and develop resources for BPCnet.org.

Briana Scurry serves as the Diversity, Equity, and Inclusion Coordinator for the Department of Computer Science at the University of North Carolina at Chapel Hill. There, she was responsible for the oversight of all DEI initiatives and communication within the department, serving as a resource for students, staff, and faculty. She also led the department community outreach and other initiatives designed to increase the representation and advocacy of historically resilient groups within CS. She will soon begin as the Assistant Director for Equity and Inclusion for the City of Raleigh.

Colleen Lewis is an Associate Professor of CS at the University of Illinois Urbana-Champaign. She researches equitable and efficient teaching practices and curates CSTeachingTips.org, a NSF-sponsored project for disseminating effective CS teaching practices.

Luther Tychonievich is an Teaching Associate Professor in CS at the University of Illinois Urbana-Champaign. He has taught many courses and workshops on equitable teaching and inclusive practices since 2009.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation (1830364, 2032231, 1940460).

REFERENCES

[1] Computing Research Association. [n. d.]. Example Activities to Use in BPC Plans. https://bpcnet.org/bpc-plan-activities/
[2] Computing Research Association. [n. d.]. Verified Departmental BPC Plans. https://bpcnet.org/verified-departmental-bpc-plans/
[3] Computing Research Association. [n. d.]. Writing BPC Goals. https://bpcnet.org/
[4] National Science Foundation. [n. d.]. CISE - Broadening Participation in Computing (BPC). https://www.nsf.gov/cise/bpc/
[5] Carl Wieman and Sarah Gilbert. 2014. The Teaching Practices Inventory: A New Tool for Characterizing College and University Teaching in Mathematics and Science. *CBE—Life Sciences Education* 13, 3 (Sept. 2014), 552–569. https://doi.org/10.1187/cbe.14-02-0023

Innovative Strategies for genAI in CS Courses

Jill Westerlund†
Computer Science
University of Alabama
Tuscaloosa, AL, USA
jhwesterlund@ua.edu

Sandy Czajka
Computer Science
Riverside Brookfield High School
Oak Park, IL, USA
czajkas@rbhs208.net

Andrew Kuemmel
Computer Sciences
University of Wisconsin
Madison, WI, USA
kuemmel@wisc.edu

ABSTRACT

Students are using generative artificial intelligence (genAI), organizations are embracing AI and machine learning, tools are emerging almost daily, and addressing these evolving technologies can be overwhelming. Rather than choosing to ignore genAI, instructors of computer science (CS) can find ways to teach with and guide students in the use of genAI in their courses. Teaching about genAI can be incorporated with instruction about effective and appropriate uses of the ever-growing tools.

This special session brings together three experienced CS educators who integrate genAI in their work with high school students, college students, and in-service teachers. The session environment allows for participant involvement in three model activities that showcase genAI tools with learner-focused practices. Participants will be provided supporting teaching resources for each guided activity and encouraged to discuss with peers and presenters.

CCS CONCEPTS

• Artificial intelligence • computer science education

KEYWORDS

genAI, AI, assessment, instruction

ACM Reference format:

Jill Westerlund (Reviewer), Sandy Czajka and Andrew Kuemmel. 2024. Innovative Strategies for genAI in CS Courses. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024). March 20-23, 2024, Portland, OR, USA. ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/3626253.3633407

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3633407

1 OVERALL OBJECTIVE OF THE SESSION

The pervasive use of algorithmic systems, including AI systems that utilize machine learning, for everyday tasks and in personal, corporate, and governmental decision making is becoming commonplace [1]. Therefore, teaching with genAI is to be encouraged in order to equip learners to approach genAI as a tool for words [2]. The objectives of this interactive special session are two-fold: to ground participants in the rationale for integrating genAI in CS courses and to guide participants through three model activities for using genAI within CS courses.

2 OUTLINE OF THE SESSION

We will begin the session with five minutes to describe the context of the presenters. Each presenter will then take five minutes to share their approach to integrating genAI with instruction. Learner feedback about each model activity will be incorporated throughout. Each presenter will guide participants in a 15-minute genAI activity that is relevant for a CS course. The remaining ten minutes will allow for audience questions.

3 JILL WESTERLUND

My strategy in using genAI with undergraduate students and adult learners is to focus on tool use to produce content while assessing documentation of the process including user reflection. A process-over-product approach requires the learner to spend time documenting the genAI prompting pattern experience over rote use of information gathering from the genAI tool. Students are encouraged to use genAI before, during, or after specific assignments and reflect on the usability of genAI results for assignment completion and reflection on how student work was enhanced, if at all, by the content created by the tool.

3.1 genAI Activity: Code Analysis using Alternative Approaches Prompting

This activity uses the Alternative Approaches Prompting Pattern [3] so that the user provides program code to the genAI tool then requests the tool generate alternate ways to accomplish the same post-condition. Using guided prompting with refinement, student work will yield multiple ways to solve the same problem while analyzing pros and cons of each approach.

Jill Westerlund is an instructor of computer science at the University of Alabama after having been a high school teacher and CS teacher for three decades.

4 SANDY CZAJKA

As a high school computer science teacher, one goal this year was to help my students understand how AI is affecting the world around them in addition to appropriate ways to use AI. My concern in creating activities to help students understand and realize there is more to genAI than just getting assignments done the "easy way" was that the lesson be culturally relevant and student centered. This activity could then be referred to throughout the year to help students use genAI in responsible and ethical ways.

4.1 genAI Activity: Setting Expectations for genAI Norms in a Classroom

This activity focused on helping students understand appropriate uses of AI and limitations of AI in a variety of venues at the beginning of the semester. The activity included four stations.

1. Small group discussion with instructor on how and when to use AI, and why students should not replace learning with AI.
2. Students learned about how Grammy's are establishing ***RULES*** about which categories a song can be nominated for if AI was used in any way to create the music.
3. Students listened to an audio segment on whether AI plagiarizes and brainstormed how they can be sure they do not plagiarize if using AI as a source.
4. Students analyzed the skills AI does better than humans and which skills humans do better than AI, giving examples from their educational experiences that demonstrate these skills.

An abbreviated version of this activity will be replicated in the session with adjustments for educators and larger numbers of participants.

Sandy Czajka is a high school teacher at Riverside Brookfield High School. She has taught for 32 years and is an active contributor to teacher professional development for AP CS courses.

5 ANDREW KUEMMEL

As an instructor of a course for graduate and undergraduate teaching assistants, one of my goals is to prepare my students to create and present activities that support learning in introductory CS courses. Three areas where introductory CS students need instructional support are self-assessment, writing unit tests [4], and debugging. Students in my course completed a project in which they investigated and created a genAI activity in one of the above areas.

5.1 genAI Activity

This activity uses genAI to create unit tests. The activity was created by teaching assistants and shared with students in a CS2 course through optional study groups.

Andrew Kuemmel has over 30 years of experience teaching high school computer science and currently is a Teaching Faculty at the University of Wisconsin Madison.

5 EXPECTATIONS

This engaging session is expected to provide participants with practical experience using genAI activities.

6 SUITABILITY

A special session fits well the hands-on and interactive nature of sharing experience with the emerging technology of genAI. Participants will hear from presenters' classroom experiences as well as experiment in three guided activities during the session.

REFERENCES

[1] ACM Technology Policy Council. 2022. *Statement on principles for responsible algorithmic systems.* Association for Computing Machinery Technology Policy Office, Washington DC, USA. https://www.acm.org/binaries/content/assets/public-policy/final-joint-ai-statement-update.pdf

[2] Tim Underwood. 2023. Calculators for words. *How will artificial intelligence change higher ed?* The Chronicle, 13-15. https://www.chronicle.com/article/how-will-artificial-intelligence-change-higher-ed

[3] Jules White. 2023. *Prompt engineering for ChatGPT.* Coursera. https://www.coursera.org/learn/prompt-engineering

[4] Whalley, J. and Philpott, A. A. 2011. Unit testing approach to building novice programmers' skills and confidence. *Proceedings of the Thirteenth Australasian Computing Education Conference, Perth, Australia.* https://dl.acm.org/doi/pdf/10.5555/2459936.2459950

ACM Student Research Competition

Roro8: A Fantasy Computer for Computer Architecture and Organization Education

Zeyad Ahmed, *Assiut University*
Contact: ziyad.20375769@compit.aun.edu.eg

In Learning computer architecture and low-level programming is a complex challenge for beginners. We introduce Roro8, an 8-bit fantasy computer system designed to simplify the learning process while closely mirroring real-world computer systems. Roro8 comprises an 8-bit microprocessor (RX8), memory-mapped I/O, external peripherals, and an external display (RD6448). This unique feature enables students to become familiar with datasheets of devices, a fundamental aspect of hardware programming.

The core pedagogical goal is to have students implement an emulator, simulators, assemblers, and compilers, providing a hands-on learning experience. They can create games or see their own programs running on the display, making abstract concepts tangible and engaging. Our work addresses the challenge of teaching computer architecture by offering a novel solution. Unlike conventional system-on-chip designs, Roro8 simplifies the learning process while emphasizing practical engagement. This paper outlines the problem, reviews related work, introduces Roro8, and underscores the significance of our approach in enhancing computer science education.

Keywords: Static code analysis, assessment, algorithm visualization, educational applications

DOI: https://doi.org/10.1145/3626253.3635420

Learning with the Rashomon Augmented Reality Cube (RARc)

Candy Arce, *New Mexico State University*
Angie Hernandez, *New Mexico State University*
Ruth Torres Castillo, *New Mexico State University*
Tomas Valdez, *New Mexico State University*
Raena Cota, *New Mexico State University*
Enrico Pontelli, *New Mexico State University*
Contact: candyarc@nmsu.edu

We attempted to address the challenge of effectively conveying the innovative literary device known as the Rashomon effect (tRe), using a digital tool. We opted for Augmented Reality (AR) in combination with MERGE cubes for this purpose. These cubes served as a practical medium for illustrating the complexities of tRe, a narrative technique renowned for its ability to offer multiple perspectives on a single event or storyline. This narrative device is known for its ability to create narrative uncertainty, withholding a clear and absolute truth, and instead exposing the readers to an assortment of interpretations. The Rashomon Augmented Reality Cube (RARc) was designed with two main goals in mind. Firstly, it was created to provide a vivid visual representation of the Rashomon effect. This not only makes storytelling engaging but also serves as a stepping stone for students to explore a variety of interpretations. Secondly, it was conceived as an educational tool for introducing students to key CT principles, such as abstraction, solution design, and computational innovations. By instilling these principles, students acquire problem-solving skills that are increasingly essential in our technology-driven world. This AR tool, known as the RARc, serves as a hands-on digital activity crafted for creative writers, with a particular focus on engaging female participants. It encourages them to unleash their creativity and explore the nuances of storytelling within an immersive and interactive learning experience. In this paper, we share development and implementation strategies that were utilized throughout the design of the RARc.

Keywords: AR; Storytelling; Creative Writing

DOI: https://doi.org/10.1145/3626253.3635417

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Ethics vs. Abstraction: Comparing Learning Outcomes from Ethics-Integrated and Technical-Only Instruction

Noelle Brown, *University of Utah*
Contact: noelle.brown@utah.edu

In recognition of the growing need for ethics instruction in computing, many instructors incorporate ethics into their computing courses. However, such integration poses an important question: does incorporating ethics into computing topics enhance or distract from students' understanding of core technical content? To answer this question, I designed a controlled lab study, assessing technical content knowledge and ethical issue-spotting abilities among students exposed to ethics-integrated compared to technical-only instruction. 82 student participants were randomly assigned to receive instruction on K-Means Clustering in one of two groups: the treatment group learned the topic through an ethical narrative, and the control group learned the topic using only abstract, numeric examples. Comparing pre- and post-test learning gains between the groups can offer insights into the impact of ethics instruction on student learning outcomes. This study can help educators find the best ways to teach ethics while recognizing that computing instructors also need to provide thorough instruction on their core computing subject. Ultimately, these findings can help ensure our efforts to teach ethics within computing effectively support students' learning of both ethics and technical computing topics.

Keywords: computing ethics education; ethics; responsible computing

DOI: https://doi.org/10.1145/3626253.3635393

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Revolutionizing Student Engagement and Enrollment through Personalized, AI-Driven Dialog Systems in Higher Education

Alexander Fox, *Elizabethtown College*
Joshua Stoner, *Elizabethtown College*
Jingwen Wang, *Elizabethtown College*
Contact: wangjingwen@etown.edu

In this digital era, approaches for enhancing student engagement and enrollment are critical for higher education institutions. This study introduces AcademiBot, a novel AI-driven college dialog system designed to provide personalized assistance to students and serve as a strategic marketing tool for higher institutions. AcademiBot leverages advanced Large Language Models (LLMs) and customized datasets to revolutionize colleges-students interactions by offering accurate, human-like responses to students' inquiries. The system employs two interfaces: AcademiBot-Web and AcademiBot-VR. AcademiBot-Web serves as a web-based GUI, offering an easily accessible platform for students to interact with, while AcademiBot-VR creates an immersive virtual reality environment, providing a more engaging experience. This dual-interface approach caters to address various user preferences, ensuring a versatile and captivating user experience. We evaluate AcademiBot using diverse LLMs to explore optimal performance in terms of speed, accuracy, and response length. The results demonstrate AcademiBot's effectiveness in providing timely and accurate information, significantly improving user interaction in a digital context. By integrating advanced LLMs, AcademiBot underscores the potential of integrating advanced AI and LLMs in higher education. It is hoped that this study will pave the path toward a more interactive and personalized learning environment.

Keywords: Large Language Models; Dialog System; Virtual Reality; Higher Education

DOI: https://doi.org/10.1145/3626253.3635414

Enhancing American Sign Language Classification by Leveraging Hand Landmark Extraction

Benjamin Thomas Guerrieri, *The College of New Jersey*
Contact: guerrib2@tcnj.edu

This research explores a novel approach to enhance the accuracy of static American Sign Language (ASL) sign classification by employing hand landmark extraction through computer vision. Traditional Convolutional Neural Networks (CNN) often operate on raw pixel data, leading to high-dimensional feature spaces and extensive training times. In contrast, our method extracts key features as data points, significantly reducing the dimensionality of the data. We demonstrate that this hand landmark detection-based approach yields promising accuracy with shallower models such as k-Nearest Neighbors (kNN) and Logistic Regression, due to the inherently lower dimensionality of the data and the elimination of background clutter. Importantly, our approach excels particularly when dealing with relatively small datasets, a critical advantage considering the limited availability of public data in this field. Our research highlights the efficiency and effectiveness of hand landmark extraction as a preprocessing technique for ASL sign classification using the Google MediaPipe hand landmark model.

Keywords: Computer Vision; Supervised Learning; Feature Extraction; Shallow Learning; Classification; American Sign Language; Landmark Extraction

DOI: https://doi.org/10.1145/3626253.3635406

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Exploring K-12 Teachers' Trajectories of Engagement with Critically Conscious Computing

Brendan Henrique, *University of California, Berkeley*
Contact: bhenrique@berkeley.edu

The future of the computer science field is dependent on recruiting computer science educators from all demographics and backgrounds. Pre-service programs that emphasize issues of equity and justice have the potential to broaden participation in CS teacher education programs. In response to calls for a more critical CS education in teacher education programs, many universities' CS teacher education programs now emphasize discussions of the societal impact of technology and encourage students to critically interrogate computer science as they learn to program.

This study examined the learning trajectories of seven pre-service teachers enrolled in a Critically Conscious Computer Science teaching licensure program by inductively coding their responses to weekly reflections as well as lesson and unit plans. Novel to this approach was a focus on intermediary moments throughout the course rather than solely final course outcomes. Findings showed that critically conscious computing curricula encouraged teachers to wrestle with the tension of computer science as a powerful force in society. Teachers engaged in thinking expansively about what computer science can look like in their schools by considering multidisciplinary integrations, especially with elementary school teachers. This study shows the nuanced development and learning trajectories of pre-service computer science teachers and demonstrates the need to better understand how teachers engage with and apply discussions of social justice and computer science to their pedagogical practice.

Keywords: Teacher Education; Critical Consciousness; Teacher Learning

DOI: https://doi.org/10.1145/3626253.3635415

Accessing and Democratizing AI for Whom? Student Learning through an Algorithm-Centered Supply Chain Case Study

Ashish Hingle, *George Mason University*
Contact: ahingle2@gmu.edu

Questioning who has access to knowledge, skills, tools, and data becomes paramount as algorithms and the artificial intelligence (AI) systems they support find widespread applications. To address these concerns, "AI democratization" has become a prominent goal. In broad strokes, democratization allows more people to understand and work with AI, but a central question remains: for whom is AI being democratized? As the phrase can represent different meanings and stakeholders, grounding the concept of democratization for undergraduate students can be challenging. This ongoing work explores student engagement with the definitions of democratizing AI through a case study highlighting a fictional (but realistic) isolated community at risk of losing its last local grocery store and the potential for technology to address the supply chain fallout. Using role-play as the instructional activity for participants to engage in a collaborative peer-learning environment, students were immersed in a verisimilar discussion, envisioning the forms democratization can take. Seventy students participated in a course focused on technology's global and social impact, and their responses were analyzed before and after participating in the role-play activity. Overall, by being primed to the discussion on democratization through pre-assigned resources, students highlighted a nuanced understanding for whom democratization was relevant even before participating in the role-play. Results from statistical analysis showed significant improvement in recognizing the democratization of AI development, profits, and governance. Additionally, most students highlighted the role-play discussion as having strengthened the concepts they highlighted initially, but some described a broader view and recognition of other levels of democratization.

Keywords: Democratizing AI; Case Study; Role-Play; Algorithms; Artificial Intelligence; Supply Chain

DOI: https://doi.org/10.1145/3626253.3635405

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

The Future of the Error Message: Comparing Large Language Models and Novice Programmer Effectiveness in Fixing Errors

Brij Howard-Sarin, *Computer Science, Williams College*
Contact: bch6@williams.edu

Research on enhancing error message presentation is of great interest to teachers and developers alike because improving Integrated Development Environments (IDEs) increases early student retention and efficiency at all levels with more effective developing tools. This study aims to compare GPT-4 and novice programmer accuracy in fixing errors to assess the viability of Large Language Models as an error message enhancement tool. First, a random sample of 100,000 sessions from all users of BlueJ 5, an IDE for novice programmers, was analyzed to determine the time it took programmers to resolve coding errors. Subsequently, for each of the five most common errors, GPT-4 was given 20 randomly-selected snippets of code from Blackbox mini, a curated subset of Blackbox with source code attached, and prompted to explain and fix the errors. This study replicated prior research that proposed a Zipf-Mandelbrot Distribution of error message frequency; the five most common errors comprised 45% of all error messages. In comparing GPT-4 and novices, it was found that humans fix code at higher rates, but GPT-4 provided completely correct explanations for error messages 96% of the time. This study concludes that GPT-4 functions best as a tool to explain error messages in an interactive format, rather than as a tool to produce correct code on its own. In conclusion, GPT-4 would be best utilized to enhance the classroom experience as a chat assistant to reduce time spent on syntactical errors, leading to improved productivity and better novice retention.

Keywords: Blackbox, BlueJ, error messages, error message enhancement, GPT-4, large language models, novice programmers, IDEs

DOI: https://doi.org/10.1145/3626253.3635404

A Comparative Study of Few-Shot vs. Zero-Shot Prompting to Generate Quick and Useful Responses to Students' Periodic Reflections

Ethan Hurley, *UNC Charlotte*
Joel Okyere-Badoo, *UNC Charlotte*
Contact: j7bado@gmail.com

Our study investigates the effectiveness of leveraging Large Language Models (LLMs), such as GPT-3.5, to generate responses to student reflections. Acknowledging the intensive nature of manually handling reflections, our investigation centers on crafting prompts to automate reflection response generation. Driven by fast and meaningful response generation to student reflections, we explored both Zero-Shot learning (ZSL) and Few-Shot learning (FSL) methodologies. Our research meticulously examined the facets of each approach, highlighting the significance of consistent and meaningful responses.

The Few-Shot prompting approach involves creating a fundamental prompt based on reflection questions and desired responses, striving for consistency while facing challenges such as GPT-3.5 computational time and issues related to content "hallucinations." In contrast, Zero-Shot prompting utilizes the base prompt and response without the assistance of examples. The evaluation process entails a meticulous examination of the quality of GPT-3.5 responses compared to the original student reflections.

In the future, our study foresees integrating our devised prompting techniques as a resource for educators to promptly grasp students' learning concerns and issues. Despite challenges, Few-Shot prompting stands out as the more reliable and relevant approach, particularly in the context of email-based formats. As Machine Learning and AI continue to advance, overcoming challenges and adjusting to fluctuations in student emotions and content remains a pivotal factor in fully harnessing the capabilities of LLMs for automating the generation of responses to student reflections.

Keywords: Artificial Intelligence (AI), Large Language Models (LLMs), Zero-Shot Learning (ZSL), Few-Shot Learning (FSL), student reflections

DOI: https://doi.org/10.1145/3626253.3635400

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Recommendation System Algorithm Application to Increase Interdisciplinary Undergraduate Course Collaborations

Julianna Iannini, *The College of New Jersey*
Kyla Ramos, *The College of New Jersey*
Joe Petrecca, *The College of New Jersey*
Contact: ianninj1@tcnj.edu

A problem within many institutions is the lack of resources to create meaningful interdisciplinary partnerships together. Addressing this issue is imperative for enhancing the quality of projects and research opportunities for faculty members. Students involved in these opportunities gain a deeper understanding of how their field can be implemented in another field. Application X is a novel web application developed to expand resources for Institution X to break the boundaries of faculty specializations to create more collaborative learning opportunities across disciplines. Faculty can connect and engage with others from various disciplines who share common interests. This platform enables faculty to establish collaborations across their courses and inspire projects, bringing together students from different backgrounds, to work with one another to benefit a community partner. Completed projects can be uploaded by faculty members, which can then be reviewed and evaluated by other users of Application X to provide resources that potentially inspire new collaborations. Application X implements principles of human computation and collective intelligence to solve this issue. Reviews can help Application X become self-sustaining and maintain the reliability of the information. This poster will delve into the design of an algorithm that recommends faculty for potential course collaborations. The unique challenge encountered in this research was how to develop an algorithm that matches faculty from dissimilar disciplines but who have similar interests outside of their own discipline. The algorithm's success in addressing the challenge allows for further enriching interdisciplinary collaborations within the college course environment.

Keywords: Human Computer Interaction, Natural Language Processing, Machine Learning, Recommendation Systems.

DOI: https://doi.org/10.1145/3626253.3635411

Creating Instructional Elementary Programming Videos for Use in an Adaptive Testing and Remediation System

Abigail Jones, *Northern Kentucky University*
Manogya Aryal, *Northern Kentucky University*
Justin Selby, *Northern Kentucky University*
Seth Adjei, *Northern Kentucky University*
Contact: jonesa122@mymail.nku.edu

In computer science undergraduate programs, the first course often focuses on elementary computer programming. However, students' backgrounds can be diverse, leading to challenges in understanding the course material. We discuss our initial steps to improve elementary programming instruction by using instructional videos. A team of three content creators developed a series of instructional videos, covering forty different programming skills. These videos span various topics, from creating and running basic Python programs to understanding and creating functions. To assist elementary programming students, we plan to integrate these videos with PLACEments, an adaptive assessment and remediation system designed to help students bridge knowledge gaps. If a student answers a question incorrectly, a relevant video is provided as support. The three content creators peer reviewed the others' instructional videos according to an educational video assessment framework. These results were also standardized through a z-score to account for any reviewer biases. The initial results from the peer review are promising. However, more information will be collected as the videos are deployed through PLACEments and tested by students currently taking an elementary programming course.

Keywords: Video instruction; Python programming; Elementary programming; Undergraduate education

DOI: https://doi.org/10.1145/3626253.3635397

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Understanding Engineering Students' Ethical and Algorithmic Decision Preferences through a Consequentialist Framework

Edward Kempa, *University of Florida*
Ashish Aggarwal, *University of Florida*
Contact: kempaedward@ufl.edu

As developments in the field of artificial intelligence (AI) continue to rapidly advance its possible applications, it becomes increasingly crucial for those developing AI systems to understand how receptive the general public will be to their work. The overarching goal of this research is to understand human decision-making (HDM) and human perspectives on algorithmic decision-making based on varied payoffs and outcomes. We conducted a pair of surveys, where the participants were asked about their understanding of AI, as well as their thoughts about the application of AI in the context of an autonomous vehicle placed in an ethically challenging situation. Our analysis focuses on participants' responses to two questions characterized by experimental variations, with additional variation in the consequences presented in those same questions between the two surveys. In total, we collected 284 responses from these surveys administered to engineering students of an introductory programming course for two consecutive semesters in 2022. We qualitatively analyzed the data for individual questions using an inductive approach and identified major themes related to the question asked. From this analysis, found that engineering students' perspectives on an ethically complex scenario were greatly impacted by the controlled variance in consequences, and have developed a framework for tracing their decision-making to their decisions and reasoning. Offering valuable insight about how students reason when it comes to ethics to educators in charge of working on developing engineering ethics curricula.

Keywords: CS1; algorithmic decision making; ethics; computational reasoning; trolley problem; perception

DOI: https://doi.org/10.1145/3626253.3635421

Mining Students' Mastery Levels from CS Placements Tests via LLMs

Yi Li, *Boston College*
Riteng Zhang, *Boston College*
Danni Qu, *Boston College*
Maíra Marques Samary, *Boston College*
Contact: licds@bc.edu

In higher education, introductory Computer Science (CS) programs offer a range of foundational courses. These encompass not only the standard CS1 and CS2 courses but may also include more specialized options like CS0 and CS1.5. In order to appropriately assign students to the suitable introductory courses, many institutions utilize placement tests, which assess students' pre-existing knowledge and skills. While most institutions rely on accuracy alone to make these determinations, there is often additional information concealed within the completed tests. This paper delves into the potential of Large Language Models (LLMs) to uncover this hidden information, particularly in gaining insights into how students perform in different concepts. Moreover, our framework has the flexibility to accommodate variations in curricula across different institutions, providing additional analytical perspectives. Initially, we built a concept inventory (CI) using the concepts covered in an institution's CS0, CS1, and CS2 curricula. Next, an LLM, specifically GPT 3.5, was applied to associate each question in the placement test with one or more concepts in the CI. Finally, the results of the placement tests were scrutinized, allowing the calculation of mastery levels in each concept for individual students. These mastery levels enable institutions to gauge a student's prior knowledge across various concepts simply by using a CS placement test. Additionally, we presented a case study demonstrating the application of this framework to 267 existing placement test results at Boston College.

Keywords: Placement Test; Concept Inventory; Large Language Models; Introductory Computer Science Courses

DOI: https://doi.org/10.1145/3626253.3635403

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Simplifying Signaling Pathway Reconstruction with Containerized Random Walk Algorithms

Yancheng Liu, *Reed College*
Anna Ritz, *Reed College*
Contact: liue@reed.edu; aritz@reed.edu

Protein interactions and cellular responses are fundamental pillars of molecular systems biology. Decoding these complex signaling pathways requires advanced computational methods. One promising direction of algorithm development is using graph algorithms to identify proteins involved in signaling pathways. Despite the availability of tools, many researchers grapple with software and user experience constraints. In response, we have developed the Signaling Pathway Reconstruction Analysis Streamliner (SPRAS), a robust containerized framework that enables users to easily reconstruct signaling pathways by connecting proteins of interest within molecular interaction networks. It seamlessly integrates graph algorithms designed for pathway reconstruction with downstream visualization and clustering analysis. We contribute and integrate three random-walk-based algorithms to SPRAS, including one algorithm we developed for large networks and two other algorithms that appear in the literature. Random walk approaches have been highly successful in predicting candidate proteins involved in a signaling pathway, and integrating them into SPRAS will greatly expand the framework's ability for pathway reconstruction. We illustrate their importance by using the random walk algorithms now available in SPRAS to explore potential proteins involved in cell-cell fusion in flies. In our computational experiments, five fly proteins appeared in multiple reconstructed pathways, suggesting a potential role for them in cell-cell fusion. With the addition of these new algorithms, SPRAS will become an essential tool for unraveling the mysteries of biological interactions.

Keywords: systems biology; pathway reconstruction; computational methods; protein-protein interactions; algorithm implementation; Docker; benchmarking; algorithm integration; downstream analysis

DOI: https://doi.org/10.1145/3626253.3635412

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Energizing Web Development in the Exploring Computer Science Curriculum

Samuel B. Mazzone, *Marquette University*
Contact: samuel.mazzone@marquette.edu

The Exploring Computer Science (ECS) program, designed to increase underrepresented groups' participation in computer science (CS), adapted its web development unit in Milwaukee during the COVID-19 pandemic. The new curriculum uses the ECS Web Lab (ECS-WL) platform, which facilitates real-time HTML and CSS compilation, allowing students to create and submit web pages for teacher feedback. Each day in the curriculum, students research specific topics of interest and incorporate those ideas into the websites they develop. This hands-on approach, tailored to students' interests, aims to enhance engagement and learning effectiveness. A comprehensive assessment strategy will analyze data from ECS-WL to gauge the program's impact on inclusivity and engagement in computer science education (CSE). These findings will guide future enhancements, ensuring ECS's commitment to diversifying CSE.

Keywords: Computer Science Education; Exploring Computer Science; Web Development

DOI: https://doi.org/10.1145/3626253.3635409

Is ChatGPT the Academic Catalyst We've All been Waiting For?

Raneem Emad Mousa, *Simmons University*
Nanette Veilleux, *Simmons University*
Contact: raneemousa3@yahoo.com

The excitement around ChatGPT 3.5 underscores its potential to transform various fields in education, including STEM. However, we must approach these claims cautiously. While AI can enhance STEM education, there are ethical concerns and potential inaccuracies linked to unsupervised automated responses. To comprehensively evaluate ChatGPT's influence on STEM, we conducted a controlled experiment that involved answering a question set in mathematics and CS in a time-limited session. To avoid bias, we recruited four groups of math and CS students with similar abilities —each group comprised five students. Two groups utilized ChatGPT, while the other two did not. Students who used ChatGPT were tasked with explaining how and where they employed the tool. Conversely, students who did not use ChatGPT were asked to showcase their problem-solving process. We analyzed the responses from these four groups, alongside the analysis of ChatGPT conversations for those who employed ChatGPT. Performance, confidence level, and completion time of each participant were recorded. Experts in mathematics and CS were then consulted to review participant responses. These experts were subsequently interviewed to gain deeper insights and draw conclusive findings. Our findings show that students who didn't use ChatGPT in Mathematics scored better than those who did, Specifically, ChatGPT provided the correct working process but yielded a wrong final answer due to arithmetic mistakes. Similarly, in programming, ChatGPT led to less elegant code. Our findings provide valuable insights into the benefits and challenges of AI integration in these fields, helping educators and students to adapt to AI advancements.

Keywords: AI; ChatGPT; Education; STEM

DOI: https://doi.org/10.1145/3626253.3635398

Understanding Undergraduate Students' Participation in Computing Clubs

Brooke Nelson, *University of Florida*
Amanpreet Kapoor, *University of Florida*
Christina Gardner-McCune, *University of Florida*
Contact: kapooramanpreet@ufl.edu

Employers in the tech industry have expectations for students' involvement outside of the classroom. One avenue where students engage with computing communities of practice is student organizations or clubs. Given the lack of empirical studies on students' involvement in clubs in computing, we designed a study that aims to understand computing undergraduate students' participation in clubs and to explore what students get out of their participation in these informal learning environments. We report findings from a multi-institutional survey-based study consisting of 673 undergraduate computing students from three universities in the United States. We found that 41% of the computing students across all years participate in at least one club related to their major. Forty-seven percent of female students participated in a club compared with thirty-eight percent of male students. Through inductive content analysis, we found that students participated in six types of clubs: computing areas, professional societies, affinity groups for underrepresented students, project-based development, Greek organizations, and other types of clubs. Overall, clubs allow students to explore computing areas, develop technical and professional skills, network with industry professionals, prepare for jobs, build a community, meet and socialize with like-minded peers, and gain motivation to sustain in computing. Our work provides empirical insights into computing students' club participation and insights to our community on the importance of types of clubs that can complement students' experiences in formal education supporting students' professional development.

Keywords: clubs; student organizations; computing education

DOI: https://doi.org/10.1145/3626253.3635394

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Askers, Answerers, Non-Answerers, and Lurkers: Investigating Online Q&A Platforms Personas and Success in a CS1 Course

Nigel Newby, *University of Pennsylvania*
Contact: newbyn@seas.upenn.edu

As online platforms become more popular in higher education, students may use these platforms more frequently since they remove social and geographic barriers. However, many students don't seek help online effectively, a crucial skill for learning and developing programming knowledge. Data captured on students' help-seeking behaviors could serve as valuable information to direct feedback and just-in-time interventions for instructors. In this paper, we examined online help-seeking behaviors (the frequency and popularity of posts) of students (n=240) in a CS1 course. In addition, we analyzed the linguistic features of the students' posts on an online Q&A platform. Our study uncovered two categories of online personas: askers (students who asked questions, n=136) and lurkers (students who never posted on the online platform, n=104).

Furthermore, askers were divided into two sub-categories: answerers (askers who also answered other students' questions, n=29) and non-answerers (askers who never answered other students' questions, n=107). We assessed the association between these personas and course performance and found that askers performed significantly better than lurkers. However, there was no statistically significant difference in performance between answerers and non-answerers. When investigating the linguistic features of askers' posts, and their relationship with learning outcomes, we found that students with more readable posts (above the median readability score) performed significantly better than students who posted less readable content. Finally, we found that students with posts that received a high number of votes (above the median), and/or a high number of views (and/or unique views) performed better than the ones that did not.

Keywords: Online discussion forums; CS1; linguistic features, question asking

DOI: https://doi.org/10.1145/3626253.3635413

Enhancing Scratch Programming with Autograding

Herart Dominggus Nurue, *The University of Alabama*
Contact: hdnurue@ua.edu

Scratch is a blocks-based programming language developed by MIT that has grown in popularity due to its focus on syntax-driven construction using drag-and-drop blocks to represent a program. The Scratch interface simplifies the process of creating interactive animations, games, and stories, making it an excellent platform for teaching programming skills, especially as a first-time experience. However, assessing and grading Scratch projects manually can be a time-consuming and often subjective endeavor for educators. This student research poster introduces my preliminary work in supporting the concept of a software testing process that can offer autograding support for Scratch programs. With instant feedback from teacher-provided test cases, specific areas of program correctness can be highlighted for improvement. Students can also be required to think about their program more deeply by writing test cases before starting their program, as a type of test-driven development. Autograding is often used in higher education computer science courses as a tool to streamline and ensure objectivity in grading. My research explores the transformative potential of integrating autograding capabilities into Scratch. However, the assessment of Scratch programs can pose challenges especially when the output of a program is an image produced by a Scratch sprite. This poster presentation summarizes my ideas for creating an autograder for Scratch and how testing support can help to encourage good coding habits.

Keywords: Scratch; Software Testing; Autograding

DOI: https://doi.org/10.1145/3626253.3635407

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Cloud-Secure: An Investigation into Firefly and Grey Wolf Optimization Algorithms for Phishing Detection with Machine Learning Classifiers

Kelvin Ovabor, *The University of Alabama*
Contact: kovabor@crimson.ua.edu

This paper delves into the exploration of Firefly and Grey Wolf Optimization (GWO) algorithms, aiming to enhance phishing detection within the realm of cloud security. The primary focus lies in harnessing the potential of machine learning classifiers, such as Decision Trees, Random Forests, Support Vector Machines, and Neural Networks, and optimizing their functionality through the integration of Firefly and GWO algorithms. The research methodology spans various stages, encompassing data collection, preprocessing, feature engineering, machine learning model selection, integration of optimization algorithms, model parameter tuning, and comprehensive evaluation.

The incorporation of optimization algorithms is strategically employed to elevate feature selection and hyperparameter optimization, ultimately augmenting the classification accuracy of machine learning models. The overarching objective is to establish more resilient phishing detection systems capable of dynamically adapting to the ever-changing nature of phishing campaigns in cloud environments.

The insights derived from this research have the potential to significantly contribute to the reinforcement of cloud security, safeguarding sensitive data and fortifying organizations against the pervasive threat of phishing attacks. This study not only sheds light on the ever-evolving landscape of cybersecurity but also underscores the synergy between machine learning and optimization nature-based algorithms, emerging as a robust defense against the continuously evolving cyber menace. The findings presented in this paper offer a substantial glimpse into the intricate domain of cloud security, where the fusion of machine learning and optimization algorithms stands as a potent force against emerging cyber threats.

Keywords: Algorithms; Machine learning classifiers; Greywolf; Firefly; Phishing attacks

DOI: https://doi.org/10.1145/3626253.3635401

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Understanding Outcome Expectancy in a CS1 Course

Griffin Pitts,*University of Florida*
Ashish Aggarwal,*University of Florida*
Contact: w.pitts@ufl.edu

Within the context of computing education, we refer to outcome expectancy as students' self-estimated performance in a learning environment. We believe this construct has the potential to serve as a proxy for a broad range of motivational constructs capable of influencing students' engagement and persistence in a course. While substantial prior research has delved into other motivational factors like students' self-efficacy, there remains a need for further exploration into the nuances of students' intrinsic belief constructs in relation to their learning behavior, engagement, and performance. This paper studies the impact of factors like GPA, self-efficacy, and other identified constructs on students' outcome expectancy. We also study the group-based differences in students' performance based on outcome expectancy. For this purpose, we analyzed the data of four hundred and ten engineering students enrolled in an introductory programming course (CS1). An exploratory factor analysis of surveyed questions was performed, identifying factors linked to students' self-efficacy, attitude toward learning, and perceptions about programming. We found that students' outcome expectancy significantly differed based on each of these identified factors and GPA. Furthermore, we also found that students' performance in the course significantly differed based on their outcome expectancy. We believe that such an analysis will provide CS educators with a better grounding to understand the underlying belief constructs that influence students' participation, persistence, and performance.

Keywords: CS1; Computing Education; Confidence; Outcome Expectancy; Programming Perceptions; Beliefs; Motivation; Self-efficacy; Performance

DOI: https://doi.org/10.1145/3626253.3635419

Edit, Run, Error, Repeat: Learning Analytics to Find the Most Improved Programming Student

Johan Snider, *Uppsala University*
Contact: johan.snider@it.uu.se

In the face of learning to program, students are often divided into two camps: those who excel and those who struggle. Through this challenge, some students manage to persevere from difficulty to ability. This study aims to identify the students who have made the most significant improvements in their programming skills, by leveraging various learning analytics and metrics. The field of learning analytics in programming has often sought to identify struggling students, utilizing metrics such as the error quotient (EQ) and time-on-task. This research distinguishes itself by taking a longitudinal approach to analyzing students' programming data collected over a nine-month period. By framing error quotient through operant conditioning and time-on-task through expectancy-value theory, we aim to root these learning analytics in established learning theory to validate their relevance. These preliminary results illustrate how we can identify students who have improved the most based on these metrics.

Keywords: Computer Science Education; Learning Analytics; Error Quotient; Time-on-Task

DOI: https://doi.org/10.1145/3626253.3635423

Undergraduate Facilitators as Role Models for Middle School Learners within an AI Summer Camp

Carly Solomon, *University of Florida*
Contact: csolomon1@ufl.edu

As Artificial Intelligence (AI) becomes ubiquitous in children's lives, it is crucial to introduce new and effective ways to teach the younger generation about AI. This study describes the experience of training undergraduate students as middle school AI summer camp facilitators. These undergraduates participated in a professional development session prior to camp for 9 days (54 hours), and then facilitated AI learning and camp activities within the two-week camp. Our findings indicate that the undergraduate facilitators benefited greatly from this experience, in turn providing benefits to the middle school learners. The undergraduates developed both professional and conceptual skills. By interacting with the facilitators as "near-peers", the students learned about AI in a fun, engaging, and supportive way. These findings contribute to our understanding of how to effectively teach young learners about complex AI concepts.

Keywords: Artificial intelligence education; Middle school; Informal learning; Summer camp; Facilitator; Professional development

DOI: https://doi.org/10.1145/3626253.3635395

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Towards a Verified Implementation of a Network Stack

Caitlyn Wilde, *Lewis and Clark College*
Wyeth T. Greenlaw Rollins, *Lewis and Clark College*
Levi Overcast, *Evergreen State College*
Alain Kägi, *Lewis and Clark College*
Contact: lc21-0559@lclark.edu

We are implementing a pure IPv6 networking stack optimized to run on IoT endpoints connected to a network through an Ethernet interface. Our implementation supports only the minimal set of protocols required to enable efficient UDP-based communication. In particular, it does not include support for specialized operations such as multicasting, routing, load balancing, firewalling, network address translation, and tunneling. We are on track soon to release our network stack implementation.

Our minimal implementation delivers several important advantages: It minimizes code footprint, shortening execution paths and improving instruction cache locality, thereby improving the memory hierarchy performance. It reduces the attack surface, improving reliability and security. It eases any verification efforts. It allows the implementation to be compiled directly with the target application further improving performance by eliminating some context switches.

We are preparing to verify formally the correctness and to characterize the performance of our networking stack. We believe our implementation will provide high performance through a strict application of separation of concerns (Dijkstra, 1974) and elimination of all unnecessary data copies, while keeping the codebase amenable to verification.

Keywords: IPv6; UDP; Network stack; Formal verification; Functional correctness; Isabelle/HOL

DOI: https://doi.org/10.1145/3626253.3635416

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

A Web Development and Cloud Deployment Framework for a Software Engineering Course

Michael Wilkinson, Ariel Schechter, Bryce Lukens, Ian Wright, Joseph Cardarelli, *Worcester Polytechnic Institute*
Contact: mtwilkinson@wpi.edu

Software Engineering is an undergraduate course designed to prepare students to program in the workplace by teaching them the importance of process, analysis, and design. One approach to teaching this course is to have the students use an all-Java framework to develop an application. Java, however, is not as prevalent as it previously was, so a more modern web-based framework could have more practical applications. The goal of this project was to develop and test a web-based, cloud framework for students to use to develop an application in a Software Engineering course. To achieve this goal, the team researched a React-based framework and built a sample application using it to prove its utility. The application was then deployed on the cloud using AWS EC2 and RDS. During development, the team created a series of useful resources and collected online resources to help future students learn the technology stack quickly, allowing them to focus on software methodologies, software design, and best practices.

Keywords: Web Development; Cloud Deployment, React.js; AWS EC and RDS

DOI: https://doi.org/10.1145/3626253.3635410

SQL Query Evaluation with Large Language Model and Abstract Syntax Trees

Lili Xiang, *Khoury College of Computer Science, Northeastern University*
Contact: lilixiang.neu@gmail.com

SQL stands as the foundational language for data analysis and manipulation, playing a pivotal role in the database learning process. Proficiency in SQL is essential for students seeking to excel in data-related fields. However, the conventional approaches to assessing SQL queries rely heavily on manual grading, and the automated assessment tools are usually producing only binary decisions for the submitted queries. Our primary research objective is to develop effective methods for evaluating the quality of the SQL queries. To meet this objective, we introduce two approaches: structure-based analysis and evaluation by an instruction tuned large language model (LLM). The first approach deconstructs queries into Abstract Syntax Trees (AST) and employs cosine similarity to assess student submissions. The second approach utilizes a pre-trained LLM: FLAN-T5, fine-tuned for predicting the quality of student submissions. These methodologies are tested on a SQL dataset, and our experimental findings evaluate against a grading rubric with categories ranging from "good" to "unacceptable". The experimental results demonstrate that we can enhance the grading efficiency by applying these approaches and illustrate the ability of utilizing LLM to classify the assessed SQL statements more accurately. In addition, this research contributes to Computer Science (CS) education by integrating these approaches into our team's automated SQL statement assessment tool, improving the learning experience and evaluation process.

Keywords: SQL; Large Language model; Abstract syntax trees; CS education; Auto-grader

DOI: https://doi.org/10.1145/3626253.3635408

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Understanding Support Needs in Computer Science Labs

Weiyin Xu, *University of San Diego*
Contact: axu@sandiego.edu

In undergraduate computer science labs, ineffective collaboration and support pose challenges to comprehensive learning. The purpose of the classes is for students to be able to get real-time support as they learn to apply class concepts. Instructors and lab assistants are often left to monitor the classroom and provide support with limited time or awareness of student activities, which is exacerbated if the class takes place remotely. Recently, orchestration systems have been used to address these issues but have primarily been developed to support mathematics education. Computer science is different in that there are often multiple correct solutions and errors can occur through logical and conceptual misunderstandings, syntax, or more technical issues with the computer. In this research, I conducted 17 semi-structured interviews with students, lab assistants, and instructors to develop a holistic understanding of challenges and potential enhancements in computer science labs including beginning, intermediate, and systems programming courses, whether they take place face-to-face or remotely, to understand what effective support may look like. Using a thematic analysis of these interviews, I found that more awareness is needed around student impasses and more efficient procedures for providing feedback. These results support previous orchestration research findings. Additionally, I found that the lab classes are a way to build a sense of community by enhancing engagement and encouraging active participation, which expands our understanding of how to design computer science lab support beyond previous orchestration systems.

Keywords: Computer Science Labs; Orchestration Systems; Remote and In-person Learning

DOI: https://doi.org/10.1145/3626253.3635396

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

SIGCSE TS 2024 Workshops

Hands-on Accessibility Teaching Training for Computer Science Instructors

Catherine M. Baker, *Creighton University*
Yasmine N. Elglaly, *Western Washington Univeristy*
Kristen Shinohara, *Rochester Institute of Technology*
Kate Sonka, *Teach Access*
Rolando Mendez, *Teach Access*
Contact: catherinebaker@creighton.edu

Accessibility, the means by which a person with a disability can equitably access, engage with, and use digital technology, is an important topic for tech professions to know, but it is not commonly included in computing curricula. Prior research has shown that professional development opportunities are needed as few instructors are currently including accessibility into their courses and more instructors do not feel they have adequate training to include the information. The lack of accessibility skills in graduates is a current issue for the tech industry, which is facing a skills gap as laws and regulations place an increasing need for the skills. This workshop seeks to close that skills gap by having three instructors with extensive experience in accessibility and integrating it into their classroom partner with Teach Access, a non-profit organization with strong ties to the tech industry that focuses on bridging the accessibility skills gap across academia and industry, to share their experiences and guide instructors through making changes to their own courses. The workshop will first provide instructors with a background about accessibility. Next, we will cover examples of how to integrate accessibility into courses. Finally, instructors will work in small groups based on the courses they are teaching to develop modifications to their own courses.

Keywords: Accessibility; education; curriculum

DOI: https://doi.org/10.1145/3626253.3633430

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Visual Data Science with Blockly-DS

Luiz Barboza, *University of Memphis*
Rafael F. Mello, *Federal Rural University of Pernambuco*
Erico S. Teixeira, *CESAR*
Andrew M. Olney, *University of Memphis*
Contact: lbarboza@memphis.edu

The workshop will give educators an introduction to graphical data analysis techniques for exploring, summarizing, and effectively communicating data using Visual Blocks (Blockly) and Jupiter Notebooks. Participants will gain skills to create and interpret various graphs and plots, fostering insights into data patterns and relationships. The emphasis will be on adeptly selecting visualizations. Upon completion of this course, educators should be proficient in: Understanding the principles of graphical data analysis and their practical applications; and creating and interpreting diverse graph types and plots. The workshop will further delve into statistical methods essential for data analysis, guiding educators on employing descriptive statistics to explore data and derive meaningful business insights. The focus will be on fostering an understanding of statistical concepts and their practical application, as well as the effective interpretation and communication of findings. Additionally, the program will introduce educators to machine learning regressors, with a primary focus on linear regression. Participants will learn the skills to train, evaluate, and apply regressors to predict continuous target variables from input features. The emphasis will be on grasping and applying machine learning principles, as well as effectively interpreting and communicating model predictions. Furthermore, participants will engage in hands-on experience with machine learning classifiers, encompassing logistic regression, decision trees, naive Bayes, and neural networks. Educators will acquire the expertise to train, evaluate, and apply classifiers for predicting categorical target variables from input features. The emphasis here is on understanding and applying machine learning principles, as well as adeptly interpreting model predictions.

Keywords: Data science; IA; Machine Learning; Block-based programming

DOI: https://doi.org/10.1145/3626253.3633408

Autograding Python Code with the Pedal Framework: Feedback Beyond Unit Tests

Austin Cory Bart, *Computer Science, University of Delaware*
Luke Gusukuma, *Virginia Commonwealth University*
Contact: acbart@udel.edu

The ever-increasing enrollments in programming courses has driven the need for sophisticated grading tools that can provide students with precise, insightful, and timely feedback. This SIGCSE workshop presents an interactive session on our powerful, open-source Python autograding framework, Pedal. As a free library, Pedal is available on a wide range of grading platforms, including GradeScope, WebCAT, and BlockPy - anything that allows installation of a pure Python library. Pedal supports but goes beyond traditional unit testing, providing advanced code analysis techniques, such as type checking, liveness checking, structural code pattern matching, and more. Pedal has a large collection of assertions to evaluate dynamic program traces, query the Abstract Syntax Tree, re-execute student code under varying conditions, mock inputs, and capture outputs. The architecture enables support for novel features like question pools and integration with Large Language Models like OpenAI's GPT. Every feedback message is treated as a first-class object, empowering educators to fine-tune feedback as desired. Pedal is not just a toolset but a comprehensive pipeline optimized for feedback selection, resolution, and evaluation. With functionalities like command-line batch execution, exhaustive metadata tracking, and A/B testing, educators and researchers can analyze and refine their feedback strategies. With Pedal's successful deployment across multiple courses and institutions over the years, this workshop will offer attendees firsthand experience and a plethora of real-world examples. By the end of this workshop, participants will be proficient in leveraging Pedal, even venturing into the realm of creating interactive games and activities using the framework.

Keywords: Pedal; Python; CS1; Autograding; Autograder; Feedback; Unit Tests; Static Analysis; Dynamic Analysis; Typechecking; Wheats and Chaffs; GPT; BlockPy; Gradescope; WebCAT

DOI: https://doi.org/10.1145/3626253.3633416

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

From Informal Ed to the University – Hands-on Use of BBC Micro:bit in CS1 Courses Post-COVID

Albert Chan, *Fayetteville State University*
Tieming Geng, *Fayetteville State University*
Joseph Kabbes, *Fayetteville State University*
Mingxian Jin, *Fayetteville State University*
Longfei Wu, *Fayetteville State University*
Contact: achan@uncfsu.edu

BBC micro:bits have long been used in informal and pre-college educational environments. With a wealth of sensors and a rich ecosystem of add-ons, it provides an engaging platform for teaching basic computer science. The use of block programming separates the learning of Computational Thinking and coding from the challenges students face with language syntax. In addition, the physicality of the device increases student engagement as they see code interact with the real world.

Typical university CS1 courses are already struggling to cover the content required for CS1 and CS2 courses. Adding yet another task to the lecture component is not tenable. However, implementing the micro:bit in the lab component to complement the lecture allows the student to enjoy the engagement provided by the micro:bit in the lab to motivate learning in the lecture. This is particularly important as overall student engagement has declined post-COVID.

Creation of a simple Rock-Paper-Scissors game on the micro:bit introduces the concepts of variables and conditional statements using a block language, and allows automatic conversion to Python. Based on our experience with the use of the micro:bit in informal education settings, we know that student engagement with the device is high.

Students will shift to using the MakeCode Python block components and add functionality manually as they become more familiar with Python syntax. The workshop will provide lesson plans and assignments for a full sequence of labs to use in CS1 and how they relate to the lecture sequence.

Keywords: Micro:bit, Block Programming; CS1; MakeCode Python.

DOI: https://doi.org/10.1145/3626253.3633419

Foundational Tools for Coaching Data Storytelling

Lujie K. Chen, *University of Maryland, Baltimore County*
Jiaqi Gong, *The University of Alabama*
Louise Yarnall, *SRI International*
Contact: lujiec@umbc.edu

Data storytelling is the skill to communicate data effectively and efficiently. Effective data storytelling goes beyond data visualization and focuses on explanation with clear rhetorical functions. It starts with a set of data insights collected from the data science workflow and involves iterative and interactive processes of filtering those insights into story slices, from which data stories can be created through ordering, organizing and narration. Data storytelling is an integral component of a well-rounded data science education, which complements foundational skills like quantitative reasoning and programming. Despite its significance, solid understanding of the theory and practice of developing data storytelling competency is lacking. Data storytelling is often perceived as a mythical process where quantitative information magically transforms into compelling narratives. Designing scalable coaching tools for data storytelling requires leveraging multidisciplinary expertise from learning science, computer science, data science, communication science, and human-centered design. In this workshop, we will share some initial findings and reflections from our interdisciplinary team searching for effective coaching methods and tools to support coaching data storytelling at scale. We will present results from literature reviews and expert interviews which will be packaged into a set of foundational tools such as mental model, cognitive processes and schema for story construction, assessment strategy, as well as preliminary ideas of tools to support data storytelling coaching. We hope to use this workshop to build a community of researchers and practitioners in coaching data storytelling in postsecondary formal and informal learning context.

Keywords: data storytelling; data science education

DOI: https://doi.org/10.1145/3626253.3633432

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Decompose Graphics to Compose Programs in Python with PyTamaro

Luca Chiodini, *Università della Svizzera italiana*
Matthias Hauswirth, *Università della Svizzera italiana*
Contact: luca.chiodini@usi.ch

Programming is increasingly being taught in high schools and in non-CS majors at universities. However, the interests of this wide population are often different from those of students who decided to study computer science. Programming graphics is a great way to cater to these interests and motivate students to learn programming with examples that go beyond the more traditional "Hello World" or mathematical exercises.
In this workshop, we demonstrate how to use PyTamaro, a Python library designed to teach programming with graphics to novices. PyTamaro's design removes the need to use and explain sophisticated programming language features and directs learners' attention towards fundamental concepts of programming. The workshop combines unplugged activities and Python programming. Graphics are first created using colored paper cutouts and cards that represent PyTamaro functions; then, the composition is written as a Python program. The programming part will be carried out on the PyTamaro web platform, which also contains more than a hundred activities of varying levels of difficulty and on different themes that instructors can directly use or take inspiration from.

Keywords: Python; Graphics; Programming; Decomposition

DOI: https://doi.org/10.1145/3626253.3633422

Effective DEI Committees: Sharing and Refining Key Practice

Erika Dawson Head, *University of Massachusetts Amherst*
Colleen M. Lewis, *University of Illinois Urbana-Champaign*
Luther A. Tychonievich, *University of Illinois Urbana-Champaign*
Contact: ColleenL@illinois.edu

This workshop is intended for faculty, staff, and students engaged in Diversity, Equity, and Inclusion (DEI) work within their institution. It will build on participants' experiences in other collaborative or committee-based work and on participants' experiences with DEI. The goal of the workshop is for attendees to learn, both from the organizers and from one another, key practices for leading effective DEI activities and committees. To facilitate this discussion, the facilitators will provide a list of ten challenges faced by DEI committees: (1) recruiting and motivating contributors, (2) training contributors, (3) working with narrowly focused committee members, (4) responding to department members who resist DEI, (5) prioritizing DEI activities, (6) discovering problem situations, (7) responding to problem situations, (8) navigating power dynamics, (9) operating within legal and political constraints, and (10) running effective meetings. Participants will be invited to identify other challenges as well.

The facilitators will lead participants in discussing these challenges. Most of the workshop time will be spent in small-group discussions about these challenges with an emphasis on brainstorming solutions to them. Participants will make specific action plans for how to improve DEI work in their committees and departments.

Keywords: broadening participation in computing; diversity; equity; inclusion

DOI: http://dx.doi.org/10.1145/1122445.1122456

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Research Expansion and Infrastructure Investment for Capacity Building

Sonal Dekhane, *National Science Foundation*
Subrata Acharya, *National Science Foundation*
Contact: {sdekhane, acharyas}@nsf.gov

In this workshop National Science Foundation (NSF) program directors from two different directorates will share information about programs and funding opportunities focused on institutional and PI research capacity building. As is well known, NSF offers a wide variety of funding opportunities to promote discovery in science and engineering, accelerate technology and innovation, and advance diversity in science and engineering. This workshop aims to improve participants' knowledge about 1) capacity building programs in NSF, specifically in Computer and Information Science and Engineering (CISE) and Directorate for STEM Education (EDU) directorates, 2) writing competitive NSF proposals, and 3) designing effective project evaluation plans and budgets. The workshop will include a panel of successful principal investigators (PIs) who will share their insights on successful grant-writing and implementation. This workshop will provide an opportunity for the participants to engage with NSF program directors, an NSF evaluator, and successful PIs that could potentially help them identify appropriate funding opportunities and develop a strong NSF proposal. The workshop will include a mix of presentations, panel discussion, hands-on activities, and question and answer session.

Keywords: NSF; capacity building; funding opportunities; CISE; EDU

DOI: https://doi.org/10.1145/3626253.3633412

Develop a Competency-Based Curriculum that Purposefully Integrates Computing Skills, Cross-Disciplinary Skills, and Dispositions

Marisa E. Exter, *Purdue University*
Deepti Tagare, *Purdue University*
Mihaela Sabin, *University of New Hampshire*
Contact: mexter@purdue.edu

Competencies are knowledge, skills, and dispositions that enable professionals to successfully perform a goal-oriented task. A traditional education model focuses primarily on presenting and assessing knowledge, with student performance represented by grades. The Competency-Based Education (CBE) model focuses on each student developing and demonstrating knowledge, skills, and dispositions. This implies a difference in the approach towards curriculum (content), pedagogy (teaching methods), and assessment. This workshop will introduce basic concepts of competencies and CBE. We will present a competency list derived from research on computing professionals' experiences. Participants will develop a spiral curriculum and (re)design a course to purposefully integrate cross-disciplinary skills (e.g., communication) and dispositions (e.g., perseverance), along with computing skills. Takeaways will include: (1) an understanding of what competencies and CBE are, and what pedagogical and assessment approaches may align with CBE; (2) a document that integrates competencies across a spiral curriculum; and (3) a plan for (re)designing one of their courses. Collaborative ideation will be used to help generate ideas for each participant's unique context and goals. Higher education faculty/instructors and administrators who would like to learn more about competencies and apply CBE practices to their own program/course will find the most benefit from this workshop. Multiple people from the same program are encouraged to attend, allowing them to consider how they can plan for integrating competencies across their program-level curriculum. Please bring internet-enabled laptops/comparably sized devices, choose a course to (re)design, and have access to relevant course materials.

Keywords: Competencies; competency-based education; spiral curriculum; cross-disciplinary skills; dispositions

DOI: https://doi.org/10.1145/3626253.3633410

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Student Mission Control: Integrating Space Data Exploration into Data and Computer Science Education

Dan Garcia, *University of California, Berkeley*
Josh Hug, *University of California, Berkeley*
Anirudhan Badrinath, *Stanford University*
Michael Ball, *University of California, Berkeley*
Lauren Mock, *University of California, Berkeley*
Contact: contact-smc@berkeley.edu

Data Science (DS) blends computer science, statistics, mathematics, and visualization. The absence of a nationwide DS curriculum in the US prompts educators to independently devise engaging DS curriculum. With support from the National Science Foundation and the United States Amateur Radio on the International Space Station, UC Berkeley teaching faculty developed a pilot curriculum, Student Mission Control (SMC), to engage middle and high school students with direct access to data and experiments aboard the International Space Station (ISS) to help develop skills of data inquiry, analysis, and exploration.

SMC addresses a crucial challenge in computing education—enhancing the depth and engagement of data analysis activities with students. By using ISS on-orbit sensor data, SMC offers activities focused on data literacy, catering to an evolving need in computing courses. SMC not only enriches educational experiences, but also broadens participation in computing from underrepresented students to engage with authentic STEM-based data and visualizations, leveraging their interest in science, engineering, and space exploration.

To extend the reach of SMC, we propose a workshop at SIGCSE. This workshop will equip participants with insights into the ISS, telemetry data, SMC design principles, and curriculum highlights. Engaging participants in collaborative hands-on notebook-based lab activities, the workshop will empower educators to integrate space-themed content into their classrooms. Attendees will receive a three-hour PD certificate, supporting their efforts to innovate CS education and make it more appealing to diverse students.

Keywords: middle school; high school; curriculum; professional development; data science; data literacy; python; computer science; introductory programming; data analysis

DOI: https://doi.org/10.1145/3626253.3633417

Igniting Curiosity with BJC Sparks: A Transformative Curriculum for Middle and High School Computer Science

Daniel Garcia, *University of California, Berkeley*
Mary Fries, *Education Development Center*
Michael Ball, *University of California, Berkeley*
Lauren Mock, *University of California, Berkeley*
Contact: contact@bjc.berkeley.edu

The Beauty and Joy of Computing (BJC) Sparks curriculum is an innovative computer science curriculum designed to introduce middle school and early high school students to the world of computer science through functional programming. Developed by the creators of BJC CS Principles (CSP), BJC Sparks presents a simplified yet engaging approach to programming, emphasizing the flow of information through functions. By focusing on functions rather than commands and iteration, BJC Sparks nurtures problem-solving skills and reduces bugs in future programming experiences.

At SIGCSE 2023, the BJC Sparks team shared their insights into the curriculum's development and implementation, featuring real-world accounts from pioneering teachers. The use of microworlds and multimedia elements captured middle school students' interest, particularly in physical computing micro:bit activities. This approach showcased the potential of multimedia and data science projects in teaching functional programming through exciting, personalized activities.

Building on the success of previous summer PD workshops with 76 educators across the U.S., we propose a condensed version of our comprehensive workshop for SIGCSE. Participants will gain a deep understanding of BJC Sparks' design principles, hands-on experience with the Snap! programming language, collaborative lab activities, alignment with CSTA standards, access to teacher materials and resources, and effective pedagogical strategies. Attendees will receive a three-hour professional development certificate and will be equipped to empower the next generation of young minds in the world of functional programming and beyond.

Keywords: middle school; high school; curriculum; professional development; data science; computer science principles; introductory programming; computing education

DOI: https://doi.org/10.1145/3626253.3633415

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

A Quantum Abacus for Teaching Quantum Algorithms

Dan-Adrian German, *Indiana University*
Marcelo Pias, *Federal University of Rio Grande*
Qiao Xiang, *Xiamen University*
Contact: dgerman@indiana.edu

Today, more than sixty companies (in the world) are building quantum computers. The natural language of their quantum gates is that of linear algebra in a complex (Hilbert) vector space. Since 2017 it is known that one can replace the linear algebra with some string-rewriting rules no more complicated than the basic rules of arithmetic. The original system was introduced by Terry Rudolph and has been promoted and disseminated in large-scale outreach projects (among others) by Diana Franklin (University of Chicago), Sofia Economou and Ed Barnes (Virginia Tech) and other educators at the high-school level. In this workshop we show how a slightly modified (though still very elementary) system can be used to communicate a visual and entirely operational understanding of key quantum computation concepts such as: superposition, entanglement, phase, interference and unitary state evolution, as they occur in quantum algorithms. Examples include the phase kickback phenomenon, teleportation, and the famous Deutsch-Josza, Bernstein-Vazirani and Grover algorithms along with the GHZ game. We work out concrete examples of proving properties for quantum gates and quantum circuits without resorting at all to complex numbers or matrix multiplication; only simple, abacus-like operations are used, hence the title of the tutorial. We show how this approach can create a genuine bridge to the mathematics of quantum computation, that is, of vector and tensor algebras in complex spaces for students who may have little or no proper mathematical background.

Keywords: quantum information science; quantum computation; quantum processing unit; undergraduate curriculum; quantum architectures

DOI: https://doi.org/10.1145/3626253.3633435

AI Enhanced Learning: Powering Curated Videos with Generative Intelligence

Ananda Gunawardena, *Rutgers University, Princeton University*
Naina Chaturvedi, *Rutgers University*
Contact: andyguna@gmail.com

Instructional videos are becoming increasingly popular among computer science students. Over 78% of students frequently visit YouTube to find videos as supplement to their textbook or classroom instruction [1]. Recent surveys show that on average, 73% of students prefer having their instructors curate a supplemental video library to aid in their learning. Now, the emergence of generative AI is revolutionizing supplemental video instruction, enabling instructors to generate slides, recording scripts, and produce high-quality videos with deep search and embedded interactive activities.

Generative AI also takes the student video learning to a new level by providing AI-generated video summaries, on-demand questions, and exploration of topics in greater depth. Integrating AI into standard videos greatly expands the possibilities of video-based learning. This workshop demonstrates how educators can enhance their existing video playlists by incorporating AI to increase student engagement and establish safety measures for AI use in education. By using dynamic dashboards, scheduled content, and gamified questions, instructors can maintain student focus.

Drawing on insights from computer science courses taught at Princeton and Rutgers Universities, we will highlight the transformative potential of AI-enhanced videos in promoting active learning, particularly in large classes. We will discuss engagement strategies and real-time data visualizations applicable to any video platform. We will utilize the cubits.ai[2] platform, a Princeton University initiative that enhances the impact of computer science courses. The platform is free, and participants are encouraged to bring their own video playlists to curate them into AI-enabled collections by enhancing the student experience through integrated generative AI.

A laptop is needed to participate. Some handouts will be given.

Keywords: Instructional videos; customized videos; video summarization; AI generated content; contextualized generative AI; data-driven insights; cost-effective videos

DOI: https://doi.org/10.1145/3626253.3633418

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Creating an On-ramp to Programming for Arts and Humanities Students with Teaspoon Languages and Custom Block Languages

Mark Guzdial, *University of Michigan*
Contact: mjguz@umich.edu

Programming is a useful medium for artists and humanities majors. Learning to program provides a useful set of skills and concepts for critical computing scholars and conversational programmers. Introductory computing classes (at both secondary and post-secondary) level focus on job skills in software development or data science, which is a different focus from what arts and humanities students need. We have been developing new introductory computing courses for arts and humanities undergraduate students, explicitly including programming. The courses were designed using participatory methods with faculty who focus on creative expression and justice. The goal was to make them engaging with no requirement for prior programming background or mathematics past algebra. We use teaspoon (task-specific programming) languages and custom blocks in Snap to create a highly-scaffolded on-ramp into programming. These include supports for image processing, language recognition and generation, digital sound manipulation, creating Web pages, building chatbots, and manipulating databases. The workshop will include a presentation of the learning goals that emerged from our participatory design sessions. Participants will use our teaspoon languages and context-specific Snap! blocks during the workshop on their own laptops, and also try out the ebooks we use to help students transfer their knowledge to mainstream tools like Python, Processing, and SQL. These activities might also be useful in secondary school computing classes. (Links to all of the tools will be provided, though only some will be used hands-on in the workshop.)

Keywords: Non-majors programming; Arts; Humanities; Teaspoon languages; Block-based programming

DOI: https://doi.org/10.1145/3626253.3633414

Project-based Activities to Introduce Hardware in a Software-focused Course

Lama Hamandi, *Northeastern University*
Mark Leslie Miller, Ph.D., *Northeastern University and Learningtech.org*
Shivakumar Mathapathi, *Northeastern University*
Contact: l.hamandi@northeastern.edu

This workshop introduces attendees to the low-level components used in the design of computer hardware, allowing them to experiment with the hardware-software interface. Attendees explore hands-on experiments that are designed for students unlikely to encounter hardware topics in their course of study. These experiments are offered in bridge courses of a graduate program enrolling students without a Computer Science background at Northeastern University (the Align MSCS Program). The workshop consists of 3 groupings of hardware experiments. In one grouping, attendees use breadboarding to construct digital circuits. Transistors and resistors are used to design logic gates; then logic gates are used to design an arithmetic circuit (Half Adder); finally, two Half Adders with an OR gate are cascaded to build a larger adder. In the second grouping, attendees explore the interface between hardware and software, making software concepts tangible. Changing a bit from 0 to 1 in software causes a voltage change on a pin, causing light-emitting diodes (LEDs) to light. Operations executed on a Pico microcontroller illustrate programming concepts like ASCII representation and data structures. Additional Python constructs enable the implementation of counters and games. In the third grouping, attendees write C code to realize Linked list, Tree, and Graph implementations on Raspberry Pi (RSPi). These C programs enable hardware components by activating the hardware I/O ports of RSPi.

Keywords: Project-based learning; Computer hardware; Hardware/Software interface; Raspberry Pi; Breadboarding; Data structures; C; Python

DOI: https://doi.org/10.1145/3626253.3633425

Generative AI in Computer Science Education

Orit Hazzan, *Technion – Israel Institute of Technology*
Yael Erez, *Technion – Israel Institute of Technology Braude College of Engineering*
Contact: oritha@technion.ac.il

Generative AI has the potential to become disruptive technology for computer science education. Therefore, computer science educators must be familiar with the threats they should deal with and with the opportunities that generative-AI opens for the computer science education community.

In the workshop, we explore the integration of several generative-AI tools and applications in computer science education. Activities include lesson design, code development, test design and assessment. We address the students' and the educators' perspectives.

In addition, we explore computer science practices and soft skills to be applied with these tools as well as immediate and future applications and implications for computer science education and for the society.

At the end of the workshop, the participants will be able to use these generative AI tools in their daily educational computer science activities and beyond.

Keywords: AI; Generative AI; Computer science education; Disruptive technology; Curriculum design; Assessment; skills

DOI: https://doi.org/10.1145/3626253.3633409

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

PLCC: A Tool Set for Teaching Programming Languages Courses

Stoney Jackson, *Western New England University*
James Heliotis, *Rochester Institute of Technology (Emeritus)*
Timothy Fossum, *SUNY College at Potsdam (Emeritus)*
Contact: stoney.jackson@wne.edu

This is a hands-on laptop-recommended workshop that introduces participants to the PLCC compiler-compiler tool set for building Java-based language interpreters in an upper-level Programming Languages course. Our approach to teaching this course shows how programming language features such as language syntax, variable lifetime, procedure application, parameter passing, recursion, and object-orientation are implemented "under the hood". This workshop is designed for CS educators who want to explore how PLCC can be used to support such an approach.

A PLCC-based language is defined by a single language specification file consisting of three sections that define the language's entire lexical structure, grammar syntax, and semantics. The lexical structure specifies tokens in terms of regular expressions, grammar syntax in terms of BNF rules, and semantics in terms of Java methods. PLCC uses this definition file to create a stand-alone set of Java files that implement the behavior of the language, typically in terms of a read-eval-print loop.

The PLCC tool set consists of the plcc.py compiler-compiler written in Python3 and a set of Java support files used to build the language interpreters. The <plcc.pithon.net> website hosts the tool set along with PDF lecture notes for an entire Programming Languages course and a collection of language specifications that accompany the lecture notes. The tool set is also available in a GitHub repository.

Keywords: PLCC; Programming Languages Compiler Compiler; Programming Languages Course; Language Semantics; Tool Set

DOI: https://doi.org/10.1145/3626253.3633428

Transform Your Computer Science Course with Specifications Grading

David L. Largent, *Ball State University*
Christian Roberson, *Florida Southern College*
Linda F. Wilson, *Texas Lutheran University*
Manuel A. Pérez-Quiñones, *University of North Carolina, Charlotte*
Contact: dllargent@bsu.edu

As proposed by Linda B. Nilson in *Specifications Grading: Restoring Rigor, Motivating Students, and Saving Faculty Time*, Specifications Grading is an assessment paradigm that relies on pass/fail grading of assignments and assessments, the structuring of course content into modules linked to learning outcomes, and the bundling of assignments and assessments within those modules. One intention of this type of course grading construct is to align assessment more closely with student attainment of intended learning outcomes. Many of the features of Specifications Grading make it more equitable. While there has been very visible work in incorporating Specifications Grading in some academic areas (e.g., in mathematics), examples of the use of Specifications Grading in computer science courses are less common. The goal of this workshop is to introduce the concepts of Specifications Grading and explain how to apply these concepts to a wide range of computing courses and class sizes. Each participant should leave the workshop with the ability to revise their course syllabus and assignments to incorporate Specifications Grading. Access to many examples and resources will be provided.

Keywords: specifications grading; mastery grading; student assessment; marking; grading; efficiency; requirements; outcomes; syllabus; assessment; equity

DOI: https://doi.org/10.1145/3626253.3633426

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Using Large Language Models for Teaching Computing

Juho Leinonen, *School of Computer Science, The University of Auckland*
Stephen MacNeil, *Computing and Information Sciences, Temple University*
Paul Denny, *School of Computer Science, The University of Auckland*
Arto Hellas, *Department of Computer Science, Aalto University*
Contact: juho.leinonen@auckland.ac.nz

In the past year, large language models (LLMs) have taken the world by storm, demonstrating their potential as a transformative force in many domains including computing education. Computing education researchers have found that LLMs can solve most assessments in introductory programming courses, including both traditional code writing tasks and other popular tasks such as Parsons problems. As more and more students start to make use of LLMs, the question instructors might ask themselves is "what can I do?". We propose that one promising way forward is to integrate LLMs into teaching practice, providing all students with an equal opportunity to learn how to interact productively with LLMs as well as encounter and understand their limitations. In this workshop, we first present state-of-the-art research results on how to utilize LLMs in computing education practice, after which participants will take part in hands-on activities using LLMs. We end the workshop by brainstorming ideas with participants around adapting their classrooms to most effectively integrate LLMs while avoiding some common pitfalls.

Keywords: large language models; teaching practice; generative AI

DOI: http://dx.doi.org/10.1145/3626253.3633436

3D Java: Using Physical Objects to Make Abstract Concepts Concrete

Colleen M. Lewis, *University of Illinois at Urbana-Champaign*
Contact: ColleenL@illinois.edu

In this workshop, participants will learn to use physical Java memory models to help students develop a deep conceptual understanding of Java variables (see www.csteachingtips.org/3D). These physical models use the metaphor of a "remote control" to introduce references. Using table-top versions of the physical models, participants will discuss and be prepared to use physical objects to help students understand: (1) how primitive variables and references variables behave in similar and different ways, (2) how reference variables that both reference an object or array can both modify its content, (3) how calling a method with an argument creates a local variable within that method, and (4) inside an object method, the variable `this` references the object we called the method on. Participants will receive a set of these physical models to use in their classroom.

Keywords: notional machines; pedagogy; Java

DOI: http://dx.doi.org/10.1145/3626253.3633423

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Teaching with AI (GPT)

Rongxin Liu, *Harvard University*
Carter Zenke, *Harvard University*
Doug Lloyd, *Harvard University*
David J. Malan, *Harvard University*
Contact: malan@harvard.edu

Teaching computer science at scale can be challenging. From our experience in CS50, Harvard University's introductory course, we've seen firsthand the impactful role that generative artificial intelligence can play in education. Recognizing its potential and stakes, we integrated OpenAI's GPT into our own teaching methodology. The goal was to emulate a 1:1 teacher-to-student ratio, incorporating "pedagogical guardrails" to maintain instructional integrity. The result was a personalized, AI-powered bot in the form of a friendly rubber duck aimed at delivering instructional responses and troubleshooting without giving outright solutions. We plan to share our journey and offer insights into responsibly harnessing AI in educational settings. Participants will gain hands-on experience working with GPT through OpenAI's APIs, understanding and crafting prompts, answering questions using embedding-based search, and finally, building their own AI chatbot. Ultimately, we'll not only share lessons learned from our own approach but also equip educators hands-on with the knowledge and tools with which they, too, can implement these technologies in their unique teaching environments.

Keywords: CLI; code; code editor; command-line interface; container; Docker; editor; graphical user interface; GUI; integrated development environment; IDE; programming; text editor; terminal window; web app; web application

DOI: http://dx.doi.org/10.1145/3626253.3633433

Exploring Java Programming after Java Version 8

Shawn Lupoli, *Computer Science, Texas A&M University*
Heather Stephes, *Oracle*
Contact: slupoli@tamu.edu

We want more students to have experiences that prepare them for future careers in IT, software, and high tech industries. Students need to have a strong background in object-oriented programming that is strongly typed in preparation for working on large, collaborative software projects in real life. They need to learn modern concepts and practice applying that to today's problems. They need to be skilled in the languages that are in high demand in industry. And they need to find joy in programming to stay motivated. Modern Java is ideally suited and one of the most used languages in industry today. The education industry is primarily using Java 8. Java 8 shipped in March of 2014. It is nearly 10 years old! Java has advanced significantly since then. What are the Java programming components and capabilities most of us still using version 8 are missing out on? Recently the OpenJDK delivered Java 21 and now advances capabilities, including syntax simplifications, every 6 months. This is not your grandparents' Java. In this interactive BOF, we'd like to share relevant changes and modern capabilities in today's Java, educate the audience about the OpenJDK development process and cadence, motivate the value of staying on current versions of software such as security hygiene, solicit feedback and encourage discussion on pain points that prevent the education industry from moving ahead to current versions, and identify suggestions for supportive change.

Keywords: Java, Programming, Java Programming

DOI: https://doi.org/10.1145/3626253.3633434

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Providing Students with Standardized, Cloud-Based Programming Environments at Term's Start (for Free)

David J. Malan, *Harvard University*
Rongxin Liu, *Harvard University*
Carter Zenke, *Harvard University*
Doug Lloyd, *Harvard University*
Contact: malan@harvard.edu

For CS50 at Harvard, we have long provided students with a standardized programming environment, to avoid start-of-term technical difficulties that might otherwise arise if students had to install and configure compilers, interpreters, and debuggers on their own Macs and PCs. (For many students, "hello, world" is challenge enough on day 0, without also encountering "command not found" at the same time!) We originally provided students with shell accounts on a university-managed cluster of systems. We then transitioned to a cloud-based equivalent so as to manage the systems ourselves, root access and all. We transitioned thereafter to client-side virtual machines, to scale to more students and enable GUI-based assignments. We have since transitioned to web-based environments, complete with code tabs, terminal windows, and file explorers, initially implemented atop AWS Cloud9 and now, most recently, GitHub Codespaces, an implementation of Visual Studio (VS) Code in the cloud, free for teachers and students alike. In this workshop, we'll discuss the pedagogical and technological advantages and disadvantages of every approach and focus most of our time, hands-on, on using and configuring GitHub Codespaces itself for teaching and learning. Along the way, attendees will learn how to create their own Docker images and "devcontainers" for their own classes and any languages they teach. Attendees will learn what is possible educationally by writing their own VS Code extensions as well. And how, at term's end, to "offboard" students to VS Code itself on their own Macs and PCs, so as to continue programming independent of Codespaces.

Keywords: CLI; code; code editor; command-line interface; container; Docker; editor; graphical user interface; GUI; integrated development environment; IDE; programming; text editor; terminal window; web app; web application

DOI: http://dx.doi.org/10.1145/3626253.3633427

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Moving Beyond Game Design: Building Teacher Confidence with Scientific Modeling by Transitioning from Game Design to EcoSystem Programming

Corrina Perrone Smith, *AgentSheets, Inc.*
Nadia Repenning, *AgentSheets, Inc.*
Contact: nadia@agentsheets.com

Scalable Game Design (SGD) instruction, vetted and researched over more than a decade, enables educators with little or no programming experience to teach students to program games with AgentCubes. SGD has been used successfully with learners from upper elementary grades to high school to motivate and teach computer science and STEM topics, including sophisticated techniques such as Artificial Intelligence.

The concepts of Scalable Game Design (SGD), and the tools for Computational Thinking Patterns (CTPs), and Computational Thinking Pattern Assessment (CTPA) are directly applicable to satisfying Upper Elementary and Middle School CS and Science Standards. Common roadblocks teachers encounter with transitioning from game design to simulation programming will be explained with solutions, including classroom management tools, tutorial materials, and pedagogy will be presented.

Keywords: Scalable Game Design, STEM, Programming, CS Upper Elementary, CS Middle School, Computational Thinking, Computational Thinking Patterns, Computational Thinking Pattern Assessment, Student Motivation

DOI: https://doi.org/10.1145/3626253.3633421

Ninth SPLICE Workshop on Technology and Data Infrastructure for CS Education Research

Cliff Shaffer, *Virginia Tech*
Peter Brusilovsky, *University of Pittsburg*
Ken Koedinger, *Carnegie Mellon University*
Thomas Price, *North Carolina State University*
Tiffany Barnes, *North Carolina State University*
Behrooz Mostafavi, *North Carolina State University*
Contact: bzmostaf@ncsu.edu

Many SIGCSE attendees are either developing or using online educational tools, and all will benefit from better interoperability among these tools and better analysis of the clickstream data coming from those tools. New tools for analyzing big data leveraged by AI (e.g., deep learning for assessment) in turn improve both content and pedagogy, thus setting up a virtuous cycle fueling learning discoveries and leveraging innovation in AI: Online technologies → big data analysis → better online technologies. This NSF-supported workshop is the latest in a series of SPLICE workshops, and is a continuation of our event at SIGCSE 2023, where the SPLICE-Portal, a dedicated socio-technical research infrastructure for Computing Education Research, was presented. This year, we continue the work with several new SPLICE community working groups, including those on Dashboards, Large Language Models, Parsons Problems, and Smart Learning Content Protocols. We continue to build upon our existing collaborations developed over the course of the project to engage more members of the community in tasks that will advance the project agenda.

Keywords: Computing Education Research; Collaborative Tools; Online Technologies

DOI: https://doi.org/10.1145/3626253.3633431

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Autograding Java Assignments in Gradescope with Jacquard

Ellen Spertus, *Northeastern University*
Priscilla Jiménez, *Grinnell College*
Contact: e.spertus@northeastern.edu

Automated grading tools enable instructors to quickly assess student programming submissions in a scalable fashion. Not only do they save staff time, they enable a virtuous cycle in which students can submit work, get immediate feedback, and make improvements before the final deadline, rather than finding out about shortcomings when it is too late to get credit for fixing them. This supports mastery-based and specifications grading. Jacquard (https://github.com/espertus/jacquard) is a powerful open-source Java autograder for Gradescope that has been used by over a thousand students at two universities. Jacquard is closely integrated with other code evaluation tools, including Checkstyle (code style adherence), PMD (static analysis), JUnit 5 (unit testing), and JaCoCo (code coverage and cyclomatic complexity). To encourage students to write good tests, Jacquard supports cross-testing, where student tests are graded on their ability to find bugs in hidden instructor code while not falsely reporting bugs in correct code. Jacquard also provides a powerful API for custom tests, such as that code uses language features in the manner required by the instructor. Workshop attendees will learn how to use basic and advanced Jacquard features so they can apply Jacquard to their own assignments. Attendees will also have the opportunity to discuss how best to apply autograders in different contexts, such as in-class activities, homework assignments, and exams. A laptop with IntelliJ and Java 17 is required.

Keywords: autograding; assessment; mastery-based learning; specifications grading; computer science education

DOI: https://doi.org/10.1145/3626253.3633429

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Machine Learning on the Move: Teaching ML Kit for Firebase in a Mobile Apps Course

Todd S. Sproull, *Washington University in St. Louis*
Doug Shook, *Washington University in St. Louis*
Contact: todd@wustl.edu

This workshop will teach instructors how to incorporate Machine Learning into their mobile course using ML Kit for Firebase. ML Kit provides powerful machine learning functionality to your app running either iOS or Android and is for both experienced and novice machine learning developers. With just a few lines of code, you can use these powerful and easy to use machine learning packages. This workshop will focus on using machine learning in mobile applications to solve real-world problems.

In this workshop, we will explore some of the functionality ML Kit provides. Topics include text recognition, facial detection, recognizing points of interest, image characterization, and labeling. We will integrate a few of these features into fully functional apps running on iOS or Android. Additionally, we will explore tradeoffs between executing ML Kit in the cloud and on a device.

This workshop is targeted at instructors teaching a web development or mobile application course looking to incorporate cloud and native ML functionality. It is also suited for anyone wanting to deploy ML applications quickly using a feature-rich API. A laptop running Android Studio or Xcode is required. Prior to the workshop participants will be given instructions for software installation.

Keywords: Mobile Software Development Kit; Machine Learning Framework

DOI: https://doi.org/10.1145/3626253.3633411

Integrating Parallel and Distributed Computing in Early Computing Classes

Alan Sussman, *University of Maryland*
Sushil Prasad, *University of Texax at San Antonio*
Charles Weems, *University of Massachusetts*
Sheik Ghafoor, *Tennessee Tech University*
Ramachandran Vaidyanathan, *Louisiana State University*
Contact: asussman@umd.edu

Parallel and distributed computing (PDC) has become pervasive in all aspects of computing, so it is essential that students include parallelism and distribution in the computational thinking that they apply to problem solving, from the very beginning of their computing education. With all computing devices that students use currently having multiple cores as well as a GPU in many cases, many students' favorite applications use multiple cores and/or distributed processors. However, we are still teaching them to solve problems using only sequential thinking. Why?

This hands-on workshop will demonstrate how easy it is to open students' eyes to exploiting concurrency in problem solving. You will participate in unplugged activities that will help students to recognize examples of PDC concepts and concurrency in the world around them. You will learn how freely available libraries can be used to exploit parallelism using shared memory parallel programming, using both plugged and unplugged activities that are usable in a classroom setting.

No equipment or prior experience is required, but a laptop that can run C++, Java, and Python is recommended for following along with some code examples. The activities and discussion models have been used successfully to teach PDC concepts in early computing courses and will be available after the workshop. Participants will receive a stipend of $400 to defray their cost of registration and one-night hotel stay. The CDER center will also have a booth in the exhibition hall for additional support.

Keywords: Computing Education; PDC Education; Early Computing Class; HPC Education, Shared Memory Parallel Programming

DOI: https://doi.org/10.1145/3626253.3633424

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

SIGCSE TS 2024 Birds of a Feather

Engaging Rural Populations in Computer Science

Nathan H. Bean, *Kansas State University*
Russell Feldhausen, *Kansas State University*
Joshua L. Weese, *Kansas State University*
Michelle Friend, *University of Nebraska Omaha*
Contact: nathanhbean@ksu.edu

In the United States 1 in 5 people live in a rural area, and many other countries also have large rural populations. These rural populations face many challenges that make them less likely to engage with computer science education. Studies with specific rural populations have demonstrated that those students have less access to and engagement with educational opportunities involving computer science, and that these disparities are even greater for intersectional populations who are both rural and from a recognized group underrepresented in computer science. Moreover recent sociological research has suggested that rural populations share a complex and cohesive group identity that goes beyond "living in a rural area". While many universities and institutions are making inroads in understanding and engaging our rural populations, these efforts are often in their infancy and there is much for us to learn. In this session, we will facilitate a discussion of the challenges facing our varied rural populations and how the members of this community are seeking to address them. We invite anyone who wants to learn and/or share about ongoing efforts, identify challenges (and potential research questions), disseminate best practices, and forge new collaborations to join us.

Keywords: Computer Science Education; Outreach; K-12; PreK-12; Rural Students; Rural Populations; Culturally Relevant Pedagogy; Intersectional Identity; Rural Identity

DOI: https://doi.org/10.1145/3626253.3635365

Connecting the Computer Science Teachers Association (CSTA) Western Chapters

Lauren Bricker, *University of Washington*
Randy Macdonald, *Corvallis High School*
Sean Glantz, *CSTA/Bend Senior High School*
Contact: bricker@cs.washington.edu

This BOF is for Computer Science Teachers Association (CSTA) chapter leaders and participants from Oregon, Washington, California, and Nevada (the CSTA Western region) to come together, explore ideas for connecting and strengthening our chapters, and discuss possible opportunities for regional cooperation across chapters, such as a western regional conference. Our goals for this BOF are to build and strengthen relationships with other chapter leaders and members, share best practices from our chapters, and scope possible areas of cooperation. We'll document our shared learnings and write up a report to be sent to all regional chapters and the CSTA leadership, with the possibility of presenting our findings at the 2024 CSTA Chapter Leadership Summit to support other regions in developing similar models of regional cooperation.

After a brief introduction, we'll engage in a jigsaw/networking activity, designed to get participants to know others from different CSTA chapters and to share expertise and experiences across these communities. We'll then have a structured, group discussion regarding the value and feasibility of a regional conference within the context of the goals and activities of local chapters and the national organization. We intend to include online attendees in order to support K-12 teachers who are unable to attend conferences during a school year in person.

CSTA is an organization run by teachers, for teachers. Our goal is to continue to grow the CS community to ensure every educator has a home. Learn more at https://csteachers.org/.

Keywords: Computer Science Education, Community, Teachers, K-12, CSTA

DOI: https://doi.org/10.1145/3626253.3635370

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Primarily Undergraduate Institution Faculty

Victoria Dean, *Olin College of Engineering*
Lynn Kirabo, *Harvey Mudd College*
Sophia Krause-Levy, *University of San Diego*
Cynthia Taylor, *Oberlin College*
Contact: vdean@olin.edu

In this session, we aim to foster a community of computer science faculty across many primarily undergraduate institutions. We will foster an inclusive environment through activities that allow for multiple forms of engagement (e.g. shows of hands, small group introductions, and full group discussion). We will facilitate discussions about ways computer science is changing at PUIs and brainstorm ways that our community can be sustained outside of the SIGCSE TS.

Keywords: CS for All, Classroom Management, Undergraduate Instruction, Fostering Belonging, Faculty Development, Undergraduate Studies

DOI: https://doi.org/10.1145/3626253.3635383

Computing as a University Graduation Requirement

Zachary Dodds, *Harvey Mudd College*
Yuan Garcia, *Harvey Mudd College*
Vidushi Ojha, *University of Illinois*
Mark Guzdial, *University of Michigan*
Tamara Nelson-Fromm, *University of Michigan*
Valerie Barr, *Bard College*
Stephanis Matsumoto, *Olin College*
Contact: dodds@cs.hmc.edu

Computing is everywhere, and it's here to stay. Computing is crucial in many disciplines and influences every discipline. It's unlikely we'll willingly return to a society unmediated by computing. How do our institutions proceed?

This BoF asks, "Should computing be a requirement for all college and university students?"

Some say yes, citing potential for improving equity-of-access, for expanding students' capabilities, for diversifying the people who understand and critique computing, and for increasing the diversity of computing participation. Some say no, citing the lack of equity-of-outcomes, the infeasibility of teaching all students equitably, and students' need for freedom in choosing what they study. Some say, "Let's consider the spectrum of possibilities... ."

This session will discuss these possibilities, expressed and constrained by 2024's forces. Is computing's value saturated - or soon to be? Or is computing a meta-skill, whose practice in learning-to-learn amplifies individual efficacy along all paths? Is Computing1 too gate-kept to be as equitable a GenEd as Composition1? Or does requiring computing, in fact, help dismantle those gates? Can students adequately learn about core computing concepts via non-CS courses that use computing? What might required computing entail?

We invite and welcome all with an interest in computing-as-degree-requirement, program-requirement, or GenEd offering. The session's seed materials will highlight evidence against the idea, for the idea, and across its vast, uncertain middle. Our BoF proposers include researchers and educators, both non-CS-requiring and CS-requiring, as well as non-CS-required and CS-required "educatees." Join us!

Keywords: computing as a literacy; computing beyond cs

DOI: http://dx.doi.org/10.1145/3626253.3635389

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Experiences with Computer-Based Testing (CBT)

Armando Fox, *University of California, Berkeley*
Dan Garcia, *University of California, Berkeley*
Cinda Heeren, *University of British Columbia*
Firas Moosvi, *University of British Columbia*
Mariana Silva, *University of Illinois, Urbana-Champaign*
Matthew West, *University of Illinois, Urbana-Champaign*
Craig Zilles, *University of Illinois, Urbana-Champaign*
Contact: slupoli@tamu.edu

Affordable, secure, and scalable assessment delivery is an essential component of large university courses, whether online or in-person. The switch to Computer-Based Testing (CBT) can have a surprising, and almost transformational effect on pedagogy. Modern CBT systems provide almost unlimited flexibility in the types of questions they can support, for both manual grading and autograding, and CBT has now been adopted at several universities and is under serious consideration at others. In this bof, faculty interested in learning about CBT and how to implement it at their institution are invited to ask their questions. Faculty experienced with CBT are invited to share how CBT has changed their approach, pedagogy, and behavior and how to advocate for its adoption.

Keywords: computer-based assessment, online assessment, online testing, CBTF, computer-based testing facility

DOI: https://doi.org/10.1145/3626253.3635375

K-12 CS Teacher Education: What are We Teaching and to Whom? How do We know We are Moving the Needle?

Charity Freeman, *Computer Science Teachers Association*
Todd Lash, *University of Illinois, Urbana-Champaign*
Contact: toddlash@illinois.edu

Over a decade ago, "Running on Empty: The Failure to Teach K-12 Computer Science in the Digital Age" (Wilson et al., 2010) compelled stakeholders in the United States to recognize the critical need for K-12 computer science (CS) education. The report highlighted the absence of K-12 CS courses due to the low number and inadequate preparation of K-12 CS teachers. CS teacher education remains as a roadblock to equitable access and achievement in CS education nationally.

In response, lawmakers have taken to incorporate CS education into their curriculums, either by adopting existing policies or creating new ones (Code.org, CSTA, & ECEP, 2023). However, achieving the integration of CS into schools necessitates ongoing systemic transformation (Dettori et al., 2018; Phelps & Santo, 2022). This comprehensive educational reform requires adequately preparing CS teachers and fostering supportive school environments (Reimer, 2019). Many schools continue to face challenges in offering CS courses due to a shortage of highly qualified CS-trained teachers. Consequently, they may hire teachers with limited CS content knowledge and teaching experience, or they are unable to offer CS courses at all (Bruno & Lewis, 2022; Parker, 2023). These outcomes perpetuate inequities in who has access to high-quality CS education and student achievement (Bruno & Lewis, 2021; Ladner & Israel, 2016). Therefore, this BoF seeks to bring together researchers, teachers, and teacher education providers to better understand the ways in which CS teacher education is enacted and begin developing best practices for the field.

Keywords: K-12 CS; teacher education; broadening participation.

DOI: https://doi.org/10.1145/3626253.3635385

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Creating University CS Teacher Preparation Programs

Michelle Friend, *University of Nebraska Omaha*
Jennifer Rosato, *Northern Lights Collaborative for Computing Education*
Nathan Bean, *Kansas State University*
Russell Feldhausen, *Kansas State University*
Josh Weese, *Kansas State University*
Contact: mefriend@unomaha.edu

As K-12 computer science education grows, so does the imperative for universities to prepare highly qualified computer science teachers. An increasing number of SIGCSE participants either have developed programs at their universities to credential CS teachers in their states, or are interested in doing so. This BOF provides an opportunity to build community, share best practices, and address challenges in creating CS teacher preparation programs. We particularly welcome participants from Colleges of Education.

Keywords: teacher preparation; teacher credentialing; K-12

DOI: https://doi.org/10.1145/3626253.3635388

Integrating Computer Science in Elementary Education

Lisa Garbrecht, *The University of Texas at Austin*
Stephanie Baker, *The University of Texas at Austin*
Zhuoying Wang, *The University of Texas at Austin*
Contact: lgarbrecht@tacc.utexas.edu

There is greater demand for Computer Science (CS) to be taught in elementary education as more states pass policies requiring it. Integrating CS into elementary education provides a viable avenue to teach CS to all students and can result in more equitable outcomes. However, our research demonstrates many elementary educators have concerns about not having enough time, training, knowledge, curricula, resources, and/or support to teach or integrate CS in their general education classrooms. This session will bring together educators, curriculum developers, professional learning providers, researchers, and practitioners interested in fostering CS and computational thinking skills among young learners. Topics will include how CS is currently being taught or integrated into elementary education, reasons why teachers are or are not integrating CS, resources and supports to address barriers to teaching CS, and what is needed to increase the integration of CS in elementary education on larger scales.

Attendees will have the opportunity to share their experiences, challenges, successes, and learnings related to teaching and/or integrating CS in elementary education toward the goals of learning from diverse perspectives and building collaborations to enhance their work. The discussion will delve into the importance of and barriers to early CS education, identify best practices and inclusive resources to make it accessible and engaging, and facilitate connections to others with similar interests to build a supportive CS learning community. This session will provide a comprehensive exploration of various perspectives in CS education.

Keywords: Computer Science education; elementary education; K12 instruction; interdisciplinary connections; computing curricula

DOI: https://doi.org/10.1145/3626253.3635379

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Teaching Track Faculty in Computer Science

Olga Glebova, *University of Connecticut*
Chris Gregg, *Stanford University*
Melinda McDaniel, *Georgia Institute of Technology*
Laney Strange, *Northeastern University*
Contact: laneys@northeastern.edu

Many computer science departments have chosen to hire faculty to teach in teaching-track positions that parallel the standard tenure-track position, providing the possibility of promotion, longer-term contracts, and higher pay for excellence in teaching and service. This birds-of-a-feather is designed to gather educators, both experienced and new to teaching track positions, who are currently in such a position to share their experiences as members of the faculty of their departments and schools. This year, we plan on discussing the impacts of evaluations of teaching faculty. Student evaluations of teaching, which are known to be biased, are often weighted heavily in teaching faculty merit review, contract renewal, and the promotion process. Our discussion will encourage teaching faculty to share the extent to which student and other types of evaluations influence their career progression, how transparent expectations around evaluation outcomes are, and what they believe would be a true evaluation of teaching excellence.

Keywords: teaching faculty; teaching-track faculty; evaluation of teaching; professional development

DOI: https://doi.org/10.1145/3626253.3635361

POGIL in Computer Science for Beginners and Experts

Olga Glebova, *University of Connecticut*
Helen H. Hu, *Westminster University*
Kendra Walther, *University of Southern California*
Contact: olga.glebova@uconn.edu

Process Oriented Guided Inquiry Learning (POGIL) is a research-based instructional strategy with a proven history across STEM disciplines. In a POGIL classroom, teams of students work on activities that are specifically designed to guide them to construct their own understanding of key concepts. At the same time, students develop process skills such as communication, teamwork, problem solving, and critical thinking. POGIL incorporates practices shown to be particularly helpful for students from underrepresented populations. In a POGIL classroom, teachers are facilitators, not lecturers. Multiple studies have shown that students in POGIL classes do better on common exams and in subsequent courses. For more information, see http://cspogil.org and http://pogil.org. This BOF will (1) enable people unfamiliar with POGIL to ask questions and learn more, and (2) bring together experienced POGIL practitioners to share experiences, concerns, ideas, and insights. As in a POGIL classroom, we will discuss topics in small teams and report out to the larger group.

Keywords: Process Oriented Guided Inquiry Learning; POGIL; process skills; teamwork

DOI: https://doi.org/10.1145/3626253.3635371

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Humanitarian Open Source in the Classroom

Gregory Hislop, *Drexel University*
Heidi Ellis, *Western New England University*
Darci Burdge, *Nassau Community College*
Cameron Macdonell, *MacEwan University*
Contact: hislop@drexel.edu

Would you like to incorporate open source into your classes? Open source provides an unmatched opportunity for students to learn about large scale software, team processes, and tools that comprise professional software development. Using open source is helpful in motivating many software engineering concepts by providing scale, complexity, and on-going product evolution that is difficult to mimic in classroom-only materials. Finally, open source projects motivate students by providing an authentic, collaborative engagement environment for learning. Join us for a discussion with instructors on the myriad ways that open source software can be incorporated into classes or used to motivate students in clubs and extracurricular activities. Experienced instructors will provide examples that they've used in the classroom. Participants will be encouraged to provide examples and/or ask questions.

Humanitarian open source software provides the added benefits of appealing to students who are interested in humanitarian application of computing, and providing an opportunity to introduce all students to computing for social good. The BoF organizers are focused on humanitarian open source, but encourage faculty with interest in using any type of open source in the classroom to join in this discussion.

Keywords: HFOSS; teaching open source; computing for social good

DOI: https://doi.org/10.1145/3626253.3635373

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Equity and Inclusion Considerations in CS Education for Students Living with Mental Health and Medical Conditions

Maya Israel, *University of Florida*
Catherine Law, *Oregon State University*
Christian Murphy, *Swarthmore College*
Contact: cmurphy4@swarthmore.edu

As our community seeks to make Computer Science education more accessible amidst a global mental health crisis and with many students living with chronic illnesses and other medical conditions, it becomes more and more urgent to address student health concerns in CS classes and in the spaces that we create. Despite the important and long overdue increase in the number of diversity, equity, inclusion, and accessibility (DEIA) initiatives in CS, these efforts run the risk of overlooking the needs of students living with ongoing mental health and medical conditions, who are underrepresented and marginalized in our field.

This session seeks to explore ways in which current DEIA efforts in CS in higher education can be expanded to be more inclusive of students living with mental health and medical conditions. We seek to create a space for a constructive and open conversation in which we will discuss how current outreach efforts, recruitment and retention strategies, supportive accommodations, mentorship opportunities, community building, and career advancement can be broadened and modified to meet the unique needs of these students, while at the same time seeking to erase any stigma and misconceptions that may cause others to marginalize and exclude them. Participants will have an opportunity to discuss issues that affect CS students living with mental health and medical conditions, share resources that are available for both students and educators, and describe efforts to create and foster a culture of understanding and support for these communities within our field.

Keywords: Accessibility; Disability; Diversity

DOI: https://doi.org/10.1145/3626253.3635377

Two-Year Program Conversations: Supporting a Diversity of Students, Articulation Pathways, and More

William Kerney, *Clovis Community College*
Laura Malave, *St. Petersburg Community College*
William Pulling, *Fanshawe College*
Cara Tang, *Portland Community College*
Contact: cara.tang@pcc.edu

This Birds of a Feather (BoF) session brings together attendees interested in computing education at the associate degree (two-year) level to discuss issues of specific interest to this group of educators. Faculty members from small community colleges who may be the only instructor in computing will be able to exchange ideas with faculty members of other institutions. Members of larger departments will find like-situated individuals with similar issues to help them solve problems and make progress. All attendees will find a community of educators who understand the issues that make teaching in an associate degree program, such as at a community or technical college, different from the job at a university teaching in a baccalaureate level computing program.

The BoF facilitators will serve as discussion leaders, with prepared topics and discussion questions to keep the session flowing and productive. At the same time, the session will be open to relevant topics raised by participants. The focus on community, similar problems and solutions, resources, vision and mission that sets two-year computing programs apart will be apparent, and members will be able to return to their campuses refreshed and ready to move forward. Above all, students will be affected, and their opportunities improved because their instructors can directly apply what they discover to the current classroom.

Keywords: Community College; Two-Year College; Computing Education; Associate Degree; ACM2Y

DOI: https://doi.org/10.1145/3626253.3635378

Finding Neurodivergent Community in Computing Education

Mara Kirdani-Ryan, *University of Washignton, Seattle*
Omar Ibrahim, *Universiyt of Illinois Chicago*
Contact: marakr@uw.edu

For computing to serve humanity, all individuals must be able to feel safe within computing. While prior work has surfaced how hegemonic racial and gendered expectations manifest in computing, neurodivergent identities have received far less attention. Existing neurodiversity narratives look to deconstruct mechanisms that privilege certain ways of thinking and being over others, but those narratives are scarce within computing. Critically, narratives are constructed in dialogue with community, and prior SIGCSE conferences have largely lacked official support for this construction, relegating those conversations to corners and edges within the conference space. This BoF continues work from last SIGCSE in supporting the construction of narratives that center neurodivergent identities and creating neurodivergent communities. We have three goals for this space: 1) cultivate community around an "invisible" identity for which public disclosure is often problematic, 2) give an explicit space for folks to "unmask" within a broader conference where masking is typically expected, and 3) utilize existing work to create connections around more specific aspects of neurodivergence.

Keywords: Neurodivergent; ADHD; Autism; Disability; Masking

DOI: https://doi.org/10.1145/3626253.3635372

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Computing, Education, and Capitalism

Amy J. Ko, *University of Washington, Seattle*
Francisco Castro, *New York University*
Aakash Gautam, *University of Pittsburgh*
Anne Drew Hu, *Michigan State University*
Sara Kingsley, *Carnegie Mellon University*
Michael Lachney, *Michigan State University*
Aman Yadav, *Michigan State University*
Contact: ajko@uw.edu

Neoliberal capitalism — the favoring of privatization, marketization, and deregulation of public resources — is hegemonic in education. It shapes our institutions and everyday life, including educational practices. The increasing push for market driven computing curriculum and large tech companies' involvement in research grants and educational programs are some examples of neoliberal influences in CS education. It is not only industry and industry-backed not-for-profits such as Code.org that reify neoliberal agendas, but in many cases, our very own communities, educational institutions, and curricula. Thus, a critical and reflective examination of our educational practices is necessary. In this Birds-of-a-Feather (BoF) session, we aim to conduct reflections and discussions on the role and influence of neoliberal capitalism in computing education. We will explore 1) the influence of deregulation, 2) public-private partnerships, 3) state-subsidized corporate wealth accumulation, and 4) centralization on pedagogy, research, and academic "success". We will discover reflective questions and practices that could help us critically examine our roles and responsibilities as computing educators and professionals. Eventually, we hope to begin a larger conversation and form a community to develop collective strategies and pedagogical ideologies to resist neoliberal influences, reimagine alternatives, and repair damages caused by neoliberal computing education.

Keywords: equity, capitalism, power

DOI: https://doi.org/10.1145/3626253.3635362

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Community-based Service Learning: Best Practices in Software Projcts with Community Partners

Stan Kurkovsky, *Central Connecticut State Univesity*
Chad Williams, *Central Connecticut State Univesity*
Mikey Goldweber, *Denison University*
Nathan Sommer, *Xavier University*
Contact: mikeyg@denison.edu

Community-engaged learning is an emerging term for a wide spectrum of learning activities that bring together students and community partners in a mutually beneficial way. Service learning is a subcategory of community-engaged learning which traditionally involves the students using their skills or expertise on a project that will directly benefit the community partner. In computer science, service learning projects often include a deliverable, such as designing and building an application. Community-based service learning is service learning with a community partner, usually a non-profit, whose mission is to improve the social, environmental, or economic situation for community members.

Undertaking service learning in general and community-based service learning in particular presents significant challenges. Some of these include the identification of an appropriate non-profit community partner and project, project skill alignment, managing partner expectations (failure is an option), and the reconciliation of the variable length of meaningful service-learning projects with the fixed length of a single term. However, given how these unique learning opportunities are known to be highly beneficial to students, educators continue to pursue them in spite of their inherent difficulties.

This BoF seeks to gather together seasoned hands, novices, plus those interested in trying out community-based service learning to

- Build a community of practice? among those who undertake community-based service learning,
- share best practices,
- present innovations, and
- illuminate pitfalls and other practices to be avoided.

Keywords: Service Learning, Community-engaged Learning, Broadening Participation, Software Engineering

DOI: https://doi.org/10.1145/3626253.3635374

Disability and Accessibility in Computer Science Education

Richard E. Ladner, *University of Washington*
Brianna Blaser, *University of Washington*
Andreas Stefik, *University of Nevada, Las Vegas*
Amy J. Ko, *University of Washington*
Raja Kushalnagar, *Gallaudet University*
Contact: accesscomp@uw.edu

Students with disabilities face a variety of challenges in computer science education including those related to stigma around disability, inaccessible curriculum, instruction and tools, disability disclosure, and a lack of mentors. In addition, few computer science classes teach students to understand accessibility and design accessible technology. This BOF will bring together individuals who are interested in increasing the representation of people with disabilities in computing and improving their success as well as those interested in teaching about accessibility. Participants will share strategies to help each other do a better job of addressing disability inclusion and accessibility in our classes and research projects. Resources related to accessible tools and instruction, universal design of learning, opportunities for students, teaching accessibility, and more will be shared.

Keywords: Disability; Universal design; Accessibility; Broadening participation in computing

DOI: https:/doi.org/10.1145/3626253.3635364

BOF: Grading for Equity in Computer Science Courses

David L. Largent, *Ball State University*
Manuel A. Pérez-Quiñones, *University of North Carolina, Charlotte*
Christian Roberson, *Florida Southern College*
Linda F. Wilson, *Texas Lutheran University*
Adrienne Decker, *University at Buffalo*
Stephen H Edwards, *Virginia Tech*
Contact: dllargent@bsu.edu

The field of computer science has a problem of representation—many groups are not represented in our classroom at levels approaching their composition in society. Unfortunately, the representation issue is a larger societal issue and begins well before students enter our institutions. Though we acknowledge that building inclusive and equitable classroom environments cannot increase representation by itself, it can have an impact on retention and inclusion for members of marginalized communities.

Current grading policies overemphasize the gaming aspect of points (e.g., goal is to maximize points) in ways that distract students from paying attention to learning. Alternatives to traditional grading, such as standards or competency-based grading, specifications-based grading, and ungrading, allow instructors to change the conversation and redirect the focus on learning. The goal of this Birds of a Feather is to foster the creation of a community of like-minded educators interested in exploring alternative grading methodologies in computer science. The goal is to make computing classrooms more accessible and equitable for all students.

Keywords: Specifications Grading; Alternative Grading; Standards Based Grading; Ungrading; Equity; Computer Science Education

DOI: https://doi.org/10.1145/3626253.3635376

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Birds of a Feather Who'd Like to Share Software Together: Teaching Tools that Improve Efficiency and Outcomes

Doug Lloyd, *Harvard University*
Carter Zenke, *Harvard University*
David J. Malan, *Harvard University*
Contact: malan@harvard.edu

Odds are we've all used (or tried!) quite a few tools to facilitate efficiency inside and outside of the classroom and empower students to learn more effectively, whether on campus or off. Some of those tools are perhaps homegrown and unique to one's own institution, but freely available educational technologies abound as well, some in the cloud, some for Macs and PCs, some open-source. And quite a few commercial tools offer free or discounted educational plans as well. In this BoF, we'll begin with a whirlwind tour of the tools we ourselves use, including AI, identifying the problems they solve and how well, then quickly open the floor to everyone to share their favorites as well. Along the way, we'll jot down every tool mentioned and share the results. With educational technology an evergreen landscape, this year's list will surely be different from last! Attendees should exit this session with a better understanding of the current landscape, familiarized with innovations they can bring back to their own classes (whether high school, undergraduate, or graduate), without reinventing wheels themselves.

Keywords: educational technologies; software; tools

DOI: http://dx.doi.org/10.1145/3626253.3635360

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Discussing the Changing Landscape of Generative AI in Computing Education

Stephen MacNeil, *Temple University*
Juho Leinonen, *Aalto University*
Paul Denny, *The University of Auckland*
Natalie Kiesler, *DIPF Leibniz Institute for Research and Information in Education*
Arto Hellas, *Aalto University*
James Prather, *Abilene Christian University*
Brett A. Becker, *University College Dublin*
Michel Wermelinger, *The Open University*
Karen Reid, *University of Toronto*
Contact: stephen.macneil@temple.edu

In a previous Birds of a Feather discussion, we delved into the nascent applications of generative AI, contemplating its potential and speculating on future trajectories. Since then, the landscape has continued to evolve revealing the capabilities and limitations of these models. Despite this progress, the computing education research community still faces uncertainty around pivotal aspects such as (1) academic integrity and assessments, (2) curricular adaptations, (3) pedagogical strategies, and (4) the competencies students require to instill responsible use of these tools. The goal of this Birds of a Feather discussion is to unravel these pressing and persistent issues with computing educators and researchers, fostering a collaborative exploration of strategies to navigate the educational implications of advancing generative AI technologies. Aligned with this goal of building an inclusive learning community, our BoF is led by globally distributed leaders to facilitate multiple coordinated discussions that can lead to a broader conversation about the role of LLMs in CS education.

Keywords: large language models; computing education; academic integrity; assessment; curriculum; pedagogy

DOI: https://doi.org/10.1145/3626253.3635369

Can We Build an Excellent Undergraduate TA Program? Fostering Collaboration and Developing Best Practices

Melinda McDaniel, *Georgia Institute of Technology*
Fisayo Omojokun, *Georgia Institute of Technology*
Mary Hudachek-Buswell, *Georgia Institute of Technology*
Contact: mcdaniel@cc.gatech.edu

As the number of students taking Computer Science courses continues to rise, the need for excellent teaching assistants (TAs), becomes increasingly important. In response, numerous CS departments have turned to their expanding pool of undergraduate students to recruit Undergraduate Teaching Assistants (UTAs) to fulfill these vital roles. These Undergraduate Teaching Assistants (UTAs) not only provide help with grading, but are responsible for development of course materials, running lab sections, and mentoring both students and newly hired UTAs. Ultimately, UTAs have a major role in affecting course climate, with a particular emphasis on fostering inclusivity, especially for underrepresented student populations. Computer Science departments struggle with the challenges of finding, hiring, and training large numbers of teaching assistants. Last year this Birds of a Feather (BoF) session brought together nearly 100 attendees who discussed the challenges faced in hiring and training effective teaching assistants. This year, the session aims to focus on the exchange of information among participants, both during and after the event, fostering a collaborative effort to establish and sustain effective UTA programs. Data collected and ideas generated will be immediately available to all.

Keywords: Teaching assistants; undergraduate student instructors; TA training; pedagogy

DOI: https://doi.org/10.1145/3626253.3635382

Building Community for Graduate Students in CS Education Research

Tamara Nelson-Fromm, *University of Michigan*
Grace Barkhuff, *Georgia Institute of Technology*
Jayne Everson, *University of Washington*
Morgan M. Fong, *University of Illinois, Urbana-Champaign*
Elijah Rivera, *Brown University*
Contact: tamaranf@umich.edu

In this Birds of a Feather (BoF) session, we aim to foster a safe and open environment that allows CS education (CSEd) graduate students to share about their graduate experiences and learn from each other about graduate school, conducting research, and navigating the inherently multi-disciplinary space of CSEd. We have two main goals: (1) create specific networking opportunities between CSEd graduate students attending SIGCSE TS (hereafter just "SIGCSE") in order to foster a community of support, and (2) elicit discussions on the triumphs and tribulations that come as part of being a CSEd-focused graduate student. We plan for all attendees to leave our session with connections to fellow CSEd graduate students from other institutions (through email and a dedicated Slack) and a greater awareness of the community available to them within CS education research.

Keywords: CS Education; Graduate Students

DOI: http://doi.org/10.1145/3626253.3635366

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Sourcing Projects for CS Capstones: Challenges and Strategies

Olufisayo Omojokun, *Georgia Institute of Technology*
KellyAnn Fitzpatrick, *Georgia Institute of Technology*
Contact: kellyann@gatech.edu

When it comes to computer science capstone courses, quality of projects is often a determining factor in learning outcomes and overall student experience. However, sourcing appropriate and interesting projects–whether devised by instructors or submitted by faculty or industry clients–can be challenging. This birds of a feather session invites educators, administrators, and even former students involved with CS capstones and other project-based courses to share some of the challenges they have encountered (e.g. client recruitment, project quality and variety, IP concerns) and potential strategies, processes, and tools for addressing these challenges.

Keywords: CS capstones; project-based learning; project recruitment

DOI: https://doi.org/10.1145/3626253.3635368

Minority Serving Institution Meet-Up

Jody Paul, *Computer Sciences, Metropolitan State University of Denver*
Contact: jody@acm.org

This is an opportunity for participants from Minority Serving Institutions (MSIs) and those interested in supporting the participation from MSIs to meet, to make new enduring connections, and to establish potential collaborations.

Keywords: MSI, Minority Serving Institution

DOI: http:// doi.org/10.1145/3626253.3635359

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Cultivating and Celebrating LGBTQ+ Community in Computing Education

Joslenne Peña, *Macalester College*
F. Megumi Kivuva *University of Washington*
Mara Kirdani-Ryan, *University of Washington*
Francisco Enrique Vicente Castro, *New York University*
Amy J. Ko, *University of Washington*
Contact: jpena@macalester.edu

This Birds-of-a-Feather (BOF) session will offer LGBTQ+ researchers and practitioners at SIGCSE TS a space for networking, sharing experiences, and being in community together. Given the consistent marginalization of the LGBTQ+ community, this BOF aims to create a community at SIGCSE of professors, students, researchers, and practitioners with interest in computing education. The BOF will provide a space for people to decompress from the long conference day through playing board games and socializing. There is also space for attendees to discuss ways computing education and SIGCSE could be more inclusive to the LGBTQ+ community.

Keywords: LGBTQ+; community; identity; critical computing

DOI: https://doi.org/10.1145/3626253.3635387

BOF: Hispanics in Computing

Brianna Posadas, *Virginia Tech*
Oscar Veliz, *Northeastern University*
Patricia Ordóñez, *University of Maryland, Baltimore County*
Manuel A. Pérez-Quiñones, *University of North Carolina, Charlotte*
Contact: perez.quinones@charlotte.edu

The Hispanics in Computing group has been around for over a decade and serves as a networking group bringing together members of this community. Members of the group organize gatherings at national conferences to strengthen the social ties that connect this small community. This Birds of a Feather seeks to bring together Hispanics and other members of the CS Education Community interested in providing support to Hispanics.

Keywords: Hispanics, Latinx, Diversity, CS Education

DOI: https://doi.org/10.1145/3626253.3635381

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Topics for an Introductory Data Science Course

Michael A. Posner, *Villanova University*
April Kerby-Helm, *Winona State University*
Contact: michael.posner@villanova.edu

Introductory data science courses are appearing at colleges, universities, and high schools around the country and the world. What topics do we cover in these courses, and how and why are these decisions made? How do we consider the background knowledge of our students and how they hope to utilize their skills after this course (whether professionally, additional courses, or as an engaged citizen)? In addition, the course is being taught by computer scientists, statisticians, business analysts, mathematicians, journalists, etc. Each of these disciplines approaches the topics differently. What upskilling is required of instructors to prepare them to integrate material from academic disciplines in which they were not trained into the course? How much, if any, cross-disciplinary collaboration, and discussion occurs or should occur in designing this course? Participants in this birds-of-a-feather will share their decision processes and choices about introductory data science courses that they teach or are designing. This includes choices made about the content as well as whether and how upskilling occurs. They will review and refine a list of current data science topics created based on national surveys of data science instructors as well as a review of curriculum guidelines. Close attention will be paid to differing language between data science instructors from different academic backgrounds. We welcome new and experienced data science instructors, educators planning on or interested in teaching such a course.

Keywords: Data Science; Undergraduate Instruction; Undergraduate Studies

DOI: https://doi.org/10.1145/3626253.3635384

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

A Town Meeting: SIGCSE Committee on Expanding the Women-in-Computing Community

Wendy Powley, *Queen's University*
Gloria Townsend, *DePauw University*
Contact: wendy.powley@queensu.ca

The SIGCSE Committee "Expanding the Women-in-Computing Community" will bring together all SIGCSE attendees interested in addressing the gender disparity in computing. Our discussion leaders connect to three of the most important organizations that relate to underrepresentation: ACM-W (Ruth Lennon), CRA-WP (Tracy Camp), and NCWIT (Lecia Barker). The forum provides an important annual meeting for a large group of people who work to increase the representation of women in computing in their separate organizations and who do not customarily have an opportunity to share ideas face-to-face.

Keywords: diversity;women in computing

DOI: https://doi.org/10.1145/3626253.3635367

Teaching Ethics In CS Programs - Questions, Models, Resources, Assessments

Vance Ricks, *Northeastern University*
Meica Magnani, *Northeastern University*
Casey Fiesler, *University of Colorado-Boulder*
Grace Barkhuff, *Georgia Institute of Technology*
Stacy Doore, *Colby College*
Alexi Brooks, *University of Wisconsin-Stout*
Michael Zimmer, *Marquette University*
Contact: v.ricks@northeastern.edu

We want to facilitate an interactive gathering of faculty from a variety of disciplines and institutions – including primary, secondary, and tertiary schools – who are interested in exchanging ideas related to the learning and teaching of ethical concepts and skills in a CS or related curriculum, and evaluating the success of those efforts. We understand "curriculum" broadly to include both course and lab instruction and out-of-class experiences. Participants will leave with a set of open-access teaching resources, guidance for how best to use those resources at their own institutions, and ideas for evaluating any courses, curriculum, or programs they develop as a result. We recommend that participants bring a laptop or table computer. We will provide some handouts.

Keywords: Active Learning; Assessment; Computers and Society; Ethics; Inclusive computing curricula and pedagogy; Internships and Co-ops, K-12 Instruction; New Interdisciplinary Programs (CS + X); Responsible Computing; Social Responsibility; Undergraduate Instruction

DOI: https://doi.org/10.1145/3626253.3635386

Directing Undergraduate Studies: Sharing Practices and Problems

Bill Siever, *Washington University in St. Louis*
Contact: bsiever@gmail.com

Directing undergraduate studies (DUS) is often undertaken by an individual (possibly department chair) or committee that lacks formal training for DUS duties. In addition, growing teaching demands are often outpacing growth in faculty, which lead to unique pressures on the DUS activities that are neither addressed by recent departmental experience nor shared among local peers. The focus of this session is common Computing DUS-specific challenges: the logistics and administration of undergraduate programs. Although the session focuses on Undergraduate Studies, those charged with direction and oversight of all types of degree programs are welcome to participate and share. Participants will be given a forum to: 1) identify some of the most common DUS activities and responsibilities; 2) share challenges, successes, and tips with other participants; and most importantly, 3) a chance to network with peers who may be able to share valuable knowledge and support beyond the session. ***Online tools will be used to organize and prioritize discussion topics, so participants should bring a laptop or mobile device.***

Keywords: Director of Undergraduate Studies; DUS; Undergraduate Studies; Administration

DOI: https://doi.org/10.1145/3626253.3635363

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Platform or Foundational Concept? Approaches to Teaching Modern Web Development

Michael C. Stewart, *James Madison University*
Michael S. Kirkpatrick, *James Madison University*
Contact: stewarmc@jmu.edu

The early web was built on static HTML or pages generated dynamically by the server. In both paradigms, the combination of HTML, CSS, and JavaScript was considered a specific platform rather than a core CS concept. In recent years, the nature of web development has shifted dramatically, emphasizing web applications that limit their dependency on server-side processing. Much of the content generation is handled in response to asynchronous execution of JavaScript. JavaScript is no longer just a language for writing event handlers; it now forms the core of client-side web development. This code must interact dynamically and asynchronously with complex backend processing systems. As such, modern web development is not as straightforward as using a server-side language to generate HTML text dynamically. It requires addressing a new computational paradigm, while also placing a greater emphasis on the user experience and accessibility. In our classes, we have observed the need to rethink how this part of our curriculum is structured. A one-semester course is no longer sufficient. We have also begun starting with JavaScript and the DOM as the native language of the web. In this birds-of-a-feather (BoF) session, we will exchange ideas on how this part of the curriculum is structured and how others approach teaching these courses.

Keywords: web development, education

DOI: https://doi.org/10.1145/3626253.3635390

Mentoring, AI, and the End of Affirmative Action: Connecting with SIGCSE Reads

Nanette Veilleux, *Simmons University*
Rebecca Bates, *Minnesota State University, Mankato*
Judy Goldsmith, *University of Kentucky*
Valerie Summet, *Rollins College*
Contact: bates@mnsu.edu

This Birds of a Feather will begin with a high-level overview of the SIGCSE Reads 2024 books and then quickly move to discussion about mentoring students in the era of large language models and ChatGPT, including how students may value the curriculum differently, how learning outcomes may change, and how we can support students and alumni/ae as they work with rapidly changing job and learning expectations. We expect that many of the sessions at SIGCSE will address the radical shifts in learning outcomes and curricular changes due to LLMs. We will not focus on the particulars of these changes, but rather on mentoring in this time with *Sister Resisters: Mentoring Black Women on Campus* by Janie Victoria Ward and Tracy L. Robinson-Wood as a resource. How do we guide our students through the curriculum upheaval triggered by shifting learning outcomes? How do we help them prepare for the new instantiation of computer science?

This BOF is the primary session for SIGCSE Reads. We encourage discussion of this year's fiction works *The Lifecycle of Software Objects* by Ted Chiang and "Dolly" by Elizabeth Bear, as well as past Reads, throughout the conference.

Keywords: Computing education; science fiction; mentoring; diversity in computing

DOI: https://doi.org/10.1145/3626253.3635380

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

SIGCSE TS 2024 Demonstrations

AntiCopyPaster: An Open-Source Ecosystem for Just-in-time Duplicate Code Refactoring

Eman Abdullah AlOmar, *Stevens Institute of Technology*
Mohamed Wiem Mkaouer, *University of Michigan-Flint*
Contact: ealomar@stevens.edu mmkaouer@umich.edu

Duplicating a code fragment is the act of copying and pasting it with or without minor modifications into another section of the code base. Despite being an intuitive practice of code reuse, duplicate code brings its own challenges to software maintenance and evolution. To combat the propagation of duplicate code, we developed a plugin for IntelliJ IDEA called AntiCopyPaster, which tracks the pasting of code fragments inside the IDE and suggests the appropriate Extract Method refactoring. Unlike the existing approaches, our tool is integrated with the development workflow, and pro-actively recommends refactorings. Since not all code fragments need to be extracted, we develop a classification model to make this decision. When a code fragment is copypasted, the plugin searches for duplicates in the currently opened file, waits for a short period of time to allow the developer to edit the code, and finally inferences the refactoring decision based on a number of features. This tool can be used by educators to control the spread of duplicate code in their student's code and raise their awareness of the existence of automated refactoring techniques that exist in the IDE. The plugin and its source code are publicly available on GitHub at https://github.com/refactorings/anti-copy-paster. The demonstration video can be found on YouTube: https://youtu.be/Y1sbfpds2Ms.

Keywords: software quality; code refactoring; plugin development

DOI: https://doi.org/10.1145/3626253.3635436

Embedding Executable Code in Slides for Introductory Programming: The Case of Interactive Code Playgrounds

Lorenzo Angeli, *University of Trento*
Luca De Menego, *Thema Optical*
Maurizio Marchese, *University of Trento*
Contact: lorenzo.angeli@unitn.it

Setting up development environments may be a complex hurdle for students who approach coding for the first time, and standardising environments among students may be an even greater challenge for teachers. Researchers and teachers tried to mitigate this issue with several solutions, all with different trade-offs, which may exclude students with limited internet access, low-powered devices, or lower computer proficiency.

In this demo, we propose *Interactive Code Playgrounds* (ICPs), web pages displayed as slides that include a front-end component enabling code execution and code output display. This architecture simplifies the installation procedure, as students need only a web browser, and gives ICPs the chance to be a more long-lasting technology, as they have minimal external dependencies. As web pages, ICPs can be distributed as links or shared as single files, re-creating the user experience of PDF slide decks. Additional benefits of ICPs include a uniform development environment improved accessibility compared to PDFs due to HTML's underlying tree structure, the inclusion of semantic tags, and text captions.

Keywords: introductory programming; front-end web technologies; slideshows; accessibility

DOI: https://doi.org/10.1145/3626253.3635426

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Snap*!* 9 — Support for Teachers and Programming with Data

Michael Ball, *UC Berkeley*
Daniel Garcia, *UC Berkeley*
Victoria Phelps, *UC Berkeley*
Yuan Garica, *Harvey Mudd College*
Contact: contact@snap.berkeley.edu

This year's Snap*!* 9 release represents 10 years since Snap*!* 4.0 was initially released as a web application. Version 9 includes hundreds of new features, focused on providing students and teachers with new cloud tools, continued development of tools for working with data, and "quality of life" enhancements.

The Snap*!*Cloud has been upgraded to support dedicated student and teacher accounts. Teachers can create accounts on behalf of their students, and manage students in their class. Everyone can also choose to "follow" users and see a feed of new projects shared with the community.

It's also never been easier to work with data! Snap! 9 adds the ability to treat lists as "dictionaries" (lists of lists can now be indexed by arbitrary alphanumeric values). A number of blocks have been made variadic (i.e., supporting a variable number of inputs), saving the user the tedium of dragging over multiple copies of the same block. These include and, or, all comparison operators, and if, which adds more intermediate else if cases when the right arrow is clicked.

Finally, Parsons Puzzles can be created with a single click. The Generate Puzzle command generates a microworld with only the blocks present in the selected sprite (and hides the answer behind a menu choice).

In this demo, we'll share ways that you can use new Snap! features in the classroom, and provide attendees with the tools to build engaging experiences for students. All materials from this demonstration will be shared at https://bjc.link/snap-sigcse24

Keywords: metaprogramming; programming paradigms; block-based programming; Snap!

DOI: https://doi.org/10.1145/3626253.3635437

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

UWCSE351 Virtual Memory Simulator

Kunchen Dai, *University of Washington*
Justin Hsia, *University of Washington*
Contact: kun02@cs.washington.edu

For many students, virtual memory serves as the introduction to abstraction in the context of software-hardware interfaces, but the sheer scale and associated complexity of virtual memory make it a difficult concept to grasp holistically. To address this, visual simulators have been developed and proven to be effective in enhancing student understanding of computer organization topics. However, a combination of limited user-friendliness, comprehensiveness, and accessibility can make existing virtual memory simulators difficult to approach for students. Thus, this demonstration will introduce the University of Washington's CSE351 Virtual Memory Simulator, an interactive web tool that lets users observe the process of address translation in a more approachable manner. It improves upon the functionality and usability of existing graphical user interface (GUI) simulators by providing more comprehensive simulation (ranging from paging in and out of disk at a high level to writing individual bytes depending on permission bits), user-friendliness (auto-highlight and auto-focuses on critical states), and user instructions and concept explanations. Furthermore, the virtual memory simulator is designed as a companion tool to the CSE351 Cache Simulator and is structured similarly, making it easy to pick up when taught in conjunction with the cache simulator. The CSE351 Virtual Memory Simulator can be integrated into virtually any computer organization or operating systems course. The demonstration will include examples of student exercises and other ways to integrate the virtual memory simulator into courses. *Note: Laptops are recommended to follow along with the demo.*

Keywords: virtual memory; simulator; educational tools; address translation

DOI: https://doi.org/10.1145/3626253.3635435

Google Colab for Teaching CS and ML

Katlyn Edwards, *Colab, Google*
Corrie Scalisi, *Colab, Google*
Julianne DeMars-Smith, *Colab, Google*
Key Lee: *Colab, Google*
Contact: katlyn@google.com

Colab is a frictionless, hosted Jupyter notebook that combines text, code, and outputs into a single document. Colab allows anybody to write and execute arbitrary python code using the latest ML accelerators (GPU/TPUs) through the browser, no setup required. It is especially well suited to machine learning, data analysis and education, and serves over 10 million active users.

Colab is used extensively for teaching computer science and machine learning, giving equitable access to expensive resources and AI/ML instruction to students around the world, regardless of background. As one professor stated: "There's an equity aspect. Not everyone has a high-end laptop. Being able to say everyone has the same computing experience and they all have access to the same resources and they can start using them right away, it allows us to find more talent randomly distributed around our student population. Colab has been the best solution so far."

Additionally, Google Colab partners with Google DeepMind to launch innovative AI coding features and models to the public, giving users the ability to author code with natural language, a much simpler experience for writing code.

We are the team who builds Colab, and would love to demo our latest features, including Google Classroom integration and AI coding using Gemini, Google's latest foundation AI model. We hope to make attendees aware of these features and have them give us feedback on their usefulness and impact on the process of teaching computer science and machine learning.

Keywords: colab ;ai; jupyter

DOI: https://doi.org/10.1145/3626253.3635432

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Algovision - An Algorithm Visualization Tool

Luděk Kučera, *Faculty of Mathematics and Pysics, Charles University*
Contact: ludek@kam.mff.cuni.cz

Algorithm visualization is an educational tool that has been used for many years to show the progress of an algorithm by updating a visual representation of a data structure. Visualizations usually cover CS1 and CS2 algorithms. However, algorithm animations and high level data structure visualizations are not yet widely used in algorithm learning.

The demo features Algovision (www.algovision.org), which seeks to overcome the problems of existing algorithm visualizations using the following design principles: Algovision is implemented in JavaScript to be immediately available on any device with a web browser; it does not offer simple visualizations, but full visual algorithm lectures; it is based on feedback from several hundreds of students and many teachers who have already used it in courses; it also includes some fairly advanced algorithms; it visualizes not only the action of the code, but also the underlying algorithmic idea; it offers a built-in guide with optional synthetic voice output.

The most important and innovative features are "playgrounds": interactive scenes, where learners can prepare their own inputs (trees, graphs, . . .) and play with operations that are later used in algorithms (e.g., rotations of binary tree edges, color flipping in red-black trees, node joining and splitting in B-trees). Playgrounds make instruction significantly more efficient and highly motivate learners. The playgrounds are advantageously used at home or in cooperating student groups.

Keywords: Algorithms, Data Structures, Visualization, JavaScript, Interactivity

DOI: https://doi.org/10.1145/3626253.3635424

Embodied Code: Creative Coding in Virtual Reality

Ryan Lay, *University of California, San Diego*
Rhea Bhutada, *University of California, San Diego*
Alejandro Lobo, *University of California, San Diego*
Robert Twomey, *University of Nebraska-Lincoln,*
Amy Eguchi, *University of California, San Diego*
Ying Choon Wu, *University of California, San Diego*
Contact: rlay@ucsd.edu

This demo presents a hands-on opportunity for users to explore a new, immersive, node-based platform for creative coding in virtual reality (VR). The Embodied Coding Environment (ECE) is a merged digital/physical workspace where coding tools, spatial representations of code, the scenes that are the output of the code, and user editing histories are co-located in a 3D space. Visual programming languages (VPL) have been used as an alternative to text-based programming since Sketchpad in 1963. However, the ECE transforms visual programming into an immersive experience, allowing users to enter the world where they are actively coding. Beyond immersion, embodied coding allows users to leverage 3D space to organize, annotate, and interact with their code in ways that would not be possible through 2D representation. Our research goals center on understanding how embodied coding can make learning to code more intuitive and elicit heightened engagement through the learning process. Demo visitors will experience embodied coding first hand through simple programming tasks (e.g., spawning objects, using arrays, creating conditional statements, and more) presented in the ECE via a Meta Quest 2 system. Presenters will guide users through the coding activities and discuss future directions. Handouts with brief descriptions of the ECE platform with links to embodiedcode.net will be available. Access to the website will allow users to login and save their projects.

Keywords: Virtual Reality; Visual Programming; Creative Coding; Spatial Computing; Immersive Learning

DOI: https://doi.org/10.1145/3626253.3635428

Providing Students with Standardized, Cloud-Based Programming Environments at Term's Start (for Free)

Rongxin Liu, *Harvard University*
Charlie Liu, *Yale University*
Carter Zenke, *Harvard University*
David J. Malan, *Harvard University*
Contact: rongxinliu@cs50.harvard.edu

CS50.dev is a cloud-based programming environment offered to students taking CS50 and other CS courses at Harvard University, both on-campus or online. Built atop GitHub Codespaces, CS50.dev simplifies the initial challenges commonly faced by students and instructors because of the complexities involved in setting up programming environments at term's start. This demo offers an in-depth exploration of CS50.dev's architecture and presents a detailed guide on customizing Docker images and development container (devcontainers) to meet the specific needs of courses within GitHub Codespaces. The demo will also provide general guidance on how to help students transition from CS50.dev to using VS Code independently on their local machines at the term's end.

Keywords: code; containers; IDE; integrated development environment; programming; text editor

DOI: https://doi.org/10.1145/3626253.3635429

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education

Rongxin Liu, *Harvard University*
Carter Zenke, *Harvard University*
Charlie Liu, *Yale University*
Andrew Holmes, *Harvard University*
Patrick Thornton, *Harvard University*
David J. Malan, *Harvard University*
Contact: rongxinliu@cs50.harvard.edu

CS50.ai is an AI-based educational tool developed and integrated into CS50 at Harvard University using large language models (LLMs), supporting both in-person and online learners. CS50.ai encapsulates a variety of AI-based tools designed to enhance students' learning by approximating a 1:1 teacher-to-student ratio. We showcase: "Explain Highlighted Code," a Visual Studio (VS) Code extension that provides just-in-time explanations of code snippets; style50, a VS Code extension that offers formatting suggestions and explanations thereof; and our "CS50 Duck," an AI-based chatbot for course-related questions, implemented both as a VS Code extension and as a standalone web application. We also demonstrate the integration of our tools into Ed, the course's discussion forum. This demo will illustrate the functionality and effectiveness of these tools as well as the pedagogical "guardrails" that we put in place to ensure secure and fair usage of these tools, while sharing insights from our own experience therewith this past summer and fall.

Keywords: AI; artificial intelligence; generative AI; large language models; LLMs

DOI: https://doi.org/10.1145/3626253.3635427

Bringing a Visual Memory Model to VS Code

Matt Schwartz, *Kansas State University*
Nathan H. Bean, *Kansas State University*
Joshua L. Weese, *Kansas State University*
Russell Feldhausen, *Kansas State University*
Contact: nathanhbean@ksu.edu

We have developed a visualization tool that displays the run-time memory state of an executing program in a fashion similar to Python Tutor as an extension to the popular VS Code IDE. Our approach combines the benefits of a diagrammatic visualization emphasizing the stack frame and heap memory model adopted by Python Tutor with a professional and extensible IDE that has made inroads into the educational community. Moreover, our tool leverages the Debug Adapter Protocol adopted by Visual Studio Code, allowing it to be used for any language supported by the IDE with minimal configuration.

This open-source tool is developed under the MIT license and hosted at https://github.com/alt-cs-lab/pedagogical-vscode.

Keywords: Computer Science Education, Development Tools, VS Code, Memory Model, Visualizations, Python Tutor

DOI: https://doi.org/10.1145/3626253.3635434

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Smart Greenhouse: Bridging Physical Computing, Plant Science, and Data Literacy

Sheikh Ahmad Shah, *Boston College*
Jaai Uday Phatak, *Boston College*
Avneet Hira, *Boston College*
Helen Zhang, *Boston College*
Mike Barnett, *Boston College*
Contact: shahst@bc.edu

In this tech demo, we will be presenting the "Smart Greenhouse" project, a transdisciplinary one, incorporating aspects of physical computing, plant science, and data literacy. In this project, students use low-cost sensors and microcontrollers (i.e., Micro:bit) to gather environmental data, with teachers guiding data analysis, visualization, and interpretation. The MakeCode platform is used for coding, which has a block-based interface, aligning with the suggestion of its efficacy in enhancing student learning and interest in coding. Furthermore, the project provides insights into environmental conditions affecting plant health, bridging the gap between computational tasks and real-world applications. For data visualization, the CODAP tool developed by the Concord Consortium is utilized, which features an easier drag-and-drop approach to create graphs. Our demonstration will encompass the collection and visualization of environmental data, coding modifications, and hands-on audience participation. Currently implemented in various educational settings, the project has witnessed teachers' growing confidence in its execution and has shown promise in bolstering students' coding skills. For this demo, participants are encouraged to bring their laptops if they wish to create or modify Micro:bit codes to test with our greenhouse setup.

Keywords: Physical Computing; Transdisciplinary learning; Data literacy

DOI: https://doi.org/10.1145/3626253.3635433

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Social Media, Ethics, and Automation: Introducing a Free Interactive Online Textbook

Kyle Thayer, *University of Washington*
Susan Notess, *Durham University*
Contact: kmthayer@uw.edu, snotess@gmail.com

Computer Science educators have made recent efforts to incorporate ethics into their introductory programming classes, as well as offer new programming classes that intersect with other fields (CS+X or X+CS) to attract new and diverse groups of students. We contribute to these efforts with a free interactive online textbook which teaches intro programming in an interdisciplinary way: Social Media, Ethics, and Automation (https://social-media-ethics-automation.github.io/book/). The interdisciplinary content of our textbook is primarily organized around social media topics, with interactive programming learning and activities (e.g., modifying bots) used to explore those social media topics. This follows an X+CS integration since CS is used to support the other topic (social media), rather than vice-versa (traditional CS+X). Ethics are incorporated throughout, starting with a diverse set of ethical frameworks, and including ethical reflections with each activity. In this demo, we will explain our pedagogical goals and strategies and then show the features and interactivity of this online textbook.

Keywords: CS; ethics; social media; introductory programming; bots; CS + X

DOI: https://doi.org/10.1145/3626253.3635431

Algot: A Visual, Hands-On Approach to Introductory Computer Science

Sverrir Thorgeirsson, *ETH Zürich*
Theo B. Weidmann, *ETH Zürich*
Sara Hooshangi, *Virginia Tech*
Contact: sverrir.thorgeirsson@inf.ethz.ch

Algot is a newly developed visual programming language that seeks to bridge the syntax-semantics gap in programming via a novel implementation of programming by demonstration. Preliminary research, which will be presented separately at SIGCSE this year, suggests that Algot may be useful for teaching foundational computer science concepts at both secondary and tertiary levels. In this proposed SIGCSE demo session, attendees will have a chance to interact with Algot and learn about its potential benefits in their own classrooms.

Keywords: visual programming; programming by demonstration; introductory computer science

DOI: https://doi.org/10.1145/3626253.3635438

An Environment for Algorithmic Essays

Michel Wermelinger, *The Open University*
Michael Snowden, *The Open University*
Contact: michel.wermelinger@open.ac.uk

We define algorithmic essays as short reports that explain and contrast at least two approaches to a computational problem. Writing and reviewing others' essays requires the higher cognitive levels of the revised Bloom taxonomy (analyse, evaluate, create) and allows students to practise professional skills like: communicating clearly a problem and its solutions; giving & receiving feedback; writing clean code using appropriate tools.

We outline the requirements for an environment that supports writing, critiquing and publishing algorithmic essays, and demonstrate one such environment: an existing cloud platform (with free education accounts) for Jupyter notebooks, the natural medium to write documents that mix text and code. We wrote open-source materials that educators can adopt and adapt to introduce algorithmic essays in their courses. The materials include sample essays, a small library, and guidance on how to write and structure essays, how to give feedback, how to lint code, and how to use the platform and the library. The latter has simple auxiliary functions that allow students to test code and measure its run-time performance for best- and worst-case inputs, without the learning curve of testing & profiling frameworks.

The demo shows how to use the platform and our materials to write essays without installing or configuring software, and how to comment on essays to help improve them. We also discuss what changes educators may need to make to adapt the materials to their courses.

Keywords: communication skills, problem solving, clean code, peer review

DOI: https://doi.org/10.1145/3626253.3635430

Let's Build a Traffic Light: Teaching Physical Computing to Children with EdBoard

Ruhan Yang, *University of Colorado Boulder*
Ellen Yi-Luen Do, *University of Colorado Boulder*
Contact: ruhan.yang@colorado.edu

EdBoard is an educational breadboard designed to facilitate early learning of physical computing in children. We designed EdBoard's curriculum based on constructionist pedagogy, and our goal is to create an engaging and interactive learning environment for children.

The EdBoard curriculum comprises three components: story booklets, circuit cards, and paper builders. These elements together foster an engaging and interactive learning experience for children. These booklets provide children with a narrative context that encourages pretend play. Circuit cards can be inserted into the EdBoard, helping children understand the positions of different components within the circuit. Paper builders empower children to create functional and interactive artifacts without the EdBoard. These hands-on activities reinforce children's understanding of physical computing concepts.

During the demo, we will guide the audience through the EdBoard's curriculum design, providing insights into how these elements come together to facilitate learning. Attendees will have the opportunity to experience EdBoard as an educational tool. Furthermore, we invite open discussions on the potential of the EdBoard platform to enhance K-12 computer science education. We believe that EdBoard could play an important role in making computer science education more accessible and engaging for children.

Keywords: Physical computing; K-12 education; children

DOI: https://doi.org/10.1145/3626253.3635425

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

SIGCSE TS 2024 Lightning Talk Abstracts

How can We Leverage Static Analysis and Large Language Models to Engage Students in Software Quality Improvement

Eman Abdullah AlOmar, *Stevens Institute of Technology*
Mohamed Wiem Mkaouer, *University of Michigan-Flint*
Contact: ealomar@stevens.edu, mmkaouer@umich.edu

Static analysis tools are frequently used to scan the source code and detect deviations from the project coding guidelines. Yet, their adoption is challenged by their high false positive rate, which makes them not suitable for students and novice developers. However, Large Language Models (LLMs), such as ChatGPT, have gained widespread popularity and usage in various software engineering tasks, including testing, code review, and program comprehension. Such models represent an opportunity to reduce the ambiguity of static analysis tools and support their adoption. Yet, the effectiveness of using static analysis (i.e., PMD) to detect coding issues, and relying on LLMs (i.e., ChatGPT) to explain and recommend fix, has not yet been explored. In this talk, we aim to shed light on our experience in teaching the use of ChatGPT to cultivate a bugfix culture and leverage LLMs to improve software quality in educational settings. We share our findings to support educators in teaching students better code review strategies, and to increase students' awareness about LLM and promote software quality in education.

Keywords: large language models, quality, computing, education

DOI: https://doi.org/10.1145/3626253.3635356

Using Natural Language Processing to Explore Instructional Change Strategies in Undergraduate Science Education Literature

Emily Bolger, *Michigan State University*
Marcos Caballero, *Michigan State University & Michigan State University*
Contact: bolgerem@msu.edu

Over ten years ago, our collaborators conducted a NSF-funded project identifying four broad categories of change strategies used to improve undergraduate STEM education through a comprehensive interdisciplinary literature review of articles from 1995 to 2008. Since this first iteration, there have been many major developments in undergraduate STEM education; particularly, the rapid development in sophisticated technology tools. These developments affect the nature of classroom instruction as well as expand the bounds on analyzing a corpus of articles. Thus, it is crucial to repeat this review to better understand the changes in STEM education from the more recent past. Our goal is to use machine learning to identify, potentially new, themes in the recent literature. We plan to compare and contrast both AI-assisted modeling and traditional, human-qualitative coding approaches in an effort to: (1) identify the benefits and faults of using AI verses human coding, and (2) portray a comprehensive story of change instruction literature from 2010. This lightning talk will describe the data extraction process and preliminary results from machine learning models. In addition to sharing the beginnings of our work, we hope to gain new perspectives and ideas from computing education scientists regarding our data representation and modeling choices.

Keywords: Natural Language Processing; Instruction Change Strategies; Undergraduate Science Education

DOI: https://doi.org/10.1145/3626253.3635341

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Registered Reports: A New Way to Publish Papers

Neil C. C. Brown, *King's College London, UK*
Contact: neil.c.c.brown@kcl.ac.uk

The traditional way to publish scientific research is to design, conduct, analyse and write-up the study before submitting to any kind of peer review. However, some issues with studies (such as the choice of data to collect) can only be fixed at the design stage. Therefore some reviewer comments (and rejections) reveal a huge waste of time: the authors cannot fix the issue without re-designing and re-running the study, so the authors', participants' and reviewers' time was wasted. *Registered Reports* are a new way to publish where the peer review is done after the design stage, but *before* the study is conducted. This allows design issues to be corrected early in the research process, when it is still possible to do so. At this point, a decision is made to reject the paper, or accept it in-principle. This means that authors can know if their study will be published before they have even begun data collection. This is beneficial for authors, and reduces incentives to fish for significant results or fudge the analysis. In this talk I will explain the registered reports process, as well as why and how authors of computer science education research studies can participate in this new way of publishing.

Keywords: Registered Reports; Computer Science Education Research

DOI: https://doi.org/10.1145/3626253.3635345

Moving forward with *LogicWriter Actual,* A Web App for Early Undergraduate Writing with Mathematical Logic

Bruce W. Char, *Drexel University*
Jeremy Johnson, *Drexel University*
Steve Earth, *Drexel University*
Contact: charbw@drexel.edu

LogicWriter Actual (https://tinyurl.com/logicwriteractual) is a web app that helps early undergraduate CS students write with symbolic logic notation (⟺, ∃, ∧, Greek letters, etc.). It designed with a quick start easy-to-use interface and is compatible with most writing programs. LogicWriter Actual is designed to minimize cognitive demands on software operation so that students can focus on mathematical writing.

LogicWriter Actual has been used in our foundational CS logic class for the past three years. Preliminary results indicate that students typically learn to use it in five minutes or less and rate it favorably. Collaboration is sought for synergistic use with scaffolded software and other instructional efforts in proof writing and mathematical exposition.

Keywords: mathematics education in computer science, learning mathematical writing, mathematical writing web app

DOI: https://doi.org/10.1145/3626253.3635339

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

A Vision for the Next 15 Years of Computing Education

Adrienne Decker, *University at Buffalo*
Mark Allen Weiss, *Florida International University*
Contact: adrienne@buffalo.edu

Recently, a workshop was convened with the goal of creating a vision for computing education for the next 15 years. The first part of the workshop involved 43 members of the computing education community meeting virtually to enumerate themes within the discipline. Those themes included: diversity, equity, inclusion, ethics, broadening participation, teaching, learning, K-12, research to practice, computing's connection to other fields, and computing education research disciplinary issues. Within each of those areas, interesting and provocative questions emerged that led the organizers to dig deeper into some hard questions for the field and probe further into where the field should be heading and what difficult questions we need to tackle. Thus, a final report that probes the issues of curricular bloat, teaching and pedagogy, infusing humanities in computing, improving pathways into the discipline, and integrating social justice into the discipline has been created. This talk will introduce the main themes of the report and point the audience towards where it is available. This material is based upon work supported by the National Science Foundation under Grant No. 2039833 and 2039848. The report and other resources can be found at https://cerfutureworkshop.org.

Keywords: Computing education research; future directions; pedagogy; social justice; curriculum; humanities

DOI: https://doi.org/10.1145/3626253.3635349

Active Learning at Large-Scale: Using Video Tutorials to Learn by Teaching

Pedro Guillermo Feijóo-García, *Georgia Institute of Technology*
Nimisha Roy, *Georgia Institute of Technology*
Contact: pfeijoogarcia@gatech.edu

In an era where digital platforms like TikTok and Instagram redefine interaction, integrating active learning in large-scale computer science (CS) courses presents both a unique challenge and opportunity. This lightning talk intends to discuss a two-step strategy piloted within a second-year large-scale (i.e., over 500 students) introductory course to software engineering at a Southeast North American university. First, we tasked CS students to create a video tutorial as their midterm exam deliverable. The exam consisted of four questions: three open-ended questions for concepts on software engineering and one diagramming question that had them transform a descriptive context into a UML domain model diagram. Next, students took part in a double-masked peer review process that had them evaluate their peers' deliverables and explanations. Overall, we intended to assess students' understanding of software engineering concepts while enabling them to reflect upon their learning processes as they taught what they learned. Also, by having them peer-review among themselves, we aimed to foster an experience to enrich students' engagement and develop feedback skills. This lightning talk is to gather feedback from the CS Education community on this instructional strategy and possibly collaborate for a more extensive study on student-created artifacts in large-scale CS courses.

Keywords: Student-created artifacts, active learning, learning by teaching, large-scale instruction

DOI: https://doi.org/10.1145/3626253.3635354

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Quantifying the Effects of Advanced Placement Computer Science Principles on Participation and Diversity in AP CS

Daniela Ganelin, *Stanford University*
Thomas S. Dee, *Stanford University*
Contact: dganelin@stanford.edu

We estimate the causal impacts of a new Advanced Placement (AP) course on participation in CS education. AP offers college-level courses, exams, and potentially college credit to U.S. secondary students. Historically, female, Black, and Hispanic students have been significantly underrepresented in AP Computer Science. A new course, CS Principles, was launched in 2016-17 to attract a more diverse group of students. Compared to the Java-focused preexisting course, CS A, CS Principles features a broader framing of CS, a creative project component in assessment, and flexibility for teachers in choosing programming languages. We assemble a dataset of annual AP exam-taking and course offerings at 294 schools in Massachusetts over fifteen years, using publicly available data from the Massachusetts education department and the College Board. Using an interactive fixed effects counterfactual estimator, we estimate that offering CS Principles increased a school's chance of having any students take AP CS exams by 25 percentage points and its count of CS exams by 24. Gains were larger for Black and Hispanic and female students. Offering CS Principles had some negative effects on CS A participation but not on other AP STEM exams. Our preliminary results suggest that elements of the CS Principles launch, including course design, curriculum development, marketing, and teacher training, can inspire other large-scale initiatives promoting engagement and diversity in STEM education. In this Lightning Talk, we hope to share our results and connect with potential collaborators interested in quantitative policy analysis in CS education.

Keywords: secondary education; Advanced Placement; equity; diversity; curriculum; causal inference; quantitative research

DOI: https://doi.org/10.1145/3626253.3635351

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Develop Innovative Virtual Reality Interaction Designs for the Learning of Computer Science Concepts and Theories

Chunming Gao, *Central Washington University*
Contact: Chunming.Gao@cwu.edu

This study aims to investigate innovative mobile virtual reality (VR) interaction designs, using affordable VR devices, for enhancing the learning of computer science concepts and theories. Smartphones have been instrumental in facilitating both augmented reality (AR) and fully immersive VR experiences with low-cost platforms. A diverse range of mobile apps empowers users to engage in AR experiences, allowing them to view virtual elements on their smartphone displays and interact with them within their real-world surroundings. On the other hand, fully immersive mobile VR experiences necessitate a VR viewer device, like Google Cardboard, equipped with a smartphone slot for enhanced immersion. This cost-effective VR platform can be operated with a single button press for interaction controls, which enable users to interact with its virtual objects presented in the immersive 3D view. However, to facilitate more complex interactions, a challenge we must address in the development of smartphone-based VR applications is the limited control options they provide. Without additional controllers, mobile VR applications rely on the movements of its head-mounted device and a single available button for user input. To overcome these control limitations, we proposed implementing a combination of head pointing, dwell selection, and various button operations, including button hold, double-button click, and triple-button click. We implemented and showcased these functions within an educational VR game focused on practicing the minimum spanning tree algorithm. We aim to design innovative mobile VR interactions to facilitate effective learning of abstract computer science concepts and theories.

Keywords: virtual reality; input methods; gamification

DOI: https://doi.org/10.1145/3626253.3635340

A Comparative Study on Student and Faculty Perceptions of Online Computing Labs

Yunfei Hou, *California State University, San Bernardino*
Miranda McIntyre, *California State University, San Bernardino*
Jesus Herrera, *California State University, San Bernardino*
Joyce Fu, *Universty of California, Riverside*
Hani Aldirawi, *California State University, San Bernardino*
Contact: hou@csusb.edu

In many disciplines, the growth of online courses was propelled by the COVID-19 pandemic, but this trend moderated as health concerns receded. This study presents a comparative analysis of students' and faculty's perceptions towards online labs in three computing-related disciplines, a year after the pandemic. Through a survey with 769 students and 20 faculty responses, we found students were overwhelmingly positive about their online lab experience – as were faculty. Students and instructors both agree that 1) the perception of online lab courses has improved since the COVID-19 pandemic, and 2) it is crucial to continue investing in technology infrastructure to enhance the quality and accessibility of both online and in-person labs. However, students and instructors disagree on two issues: 1) teamwork for lab activities and assignments (i.e., faculty tended to have a more optimistic view of online collaborative activities); and 2) modality for lab sessions (i.e., student preferences were evenly split between synchronous and asynchronous labs while faculty mostly preferred synchronous online labs). In this lightning talk, we will also discuss best practices worth promoting despite the added effort for faculty.

Keywords: Online Learning; Computer Science Laboratory; Faculty Perception; Comparative Survey

DOI: https://doi.org/10.1145/3626253.3635350

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Integrating Critical Analysis of Society and Technology into K-12 Computing Through Teacher Co-Design

Anne Drew Hu, *Michigan State University*
Aman Yadav, *Michigan State University*
Contact: annedrewhu@gmail.com

As demand for K-12 computer science (CS) education grows, we argue that most students would be best served by CS classes that not only teach computational thinking/programming, but also challenge them to critically analyze the role of technology in society. One of the main barriers to implementing this into K-12 classrooms is a lack of research on how in-service CS teachers can integrate critical pedagogy into their school context and existing curricula. This lightning talk presents a proposed method to co-design lessons with current K-12 CS teachers to integrate critical perspectives into their classrooms. Teacher participants will join a synchronous summer professional development where we will teach them critical computing content drawn from relevant books and frameworks, and collaborate with them to design or modify lessons that will engage their students in critical analysis. Teachers will be encouraged to include content relevant to their communities. For example, a teacher in Detroit might teach facial recognition in the context of continued use of the technology by police, despite wrongful arrests. Results from this work will inform future sociocultural content integration into K-12 CS courses (e.g. “ethics content”). Feedback from the audience will be used to improve the methods and literature review of the study.

Keywords: K-12 CS**;** critical theory**;** ethics**;** computing education

DOI: https://doi.org/10.1145/3626253.3635338

Scaling Responsible Computing Globally: Lessons from the US, Kenya, and India

Crystal Lee, *Massachusetts Institute of Technology*
Chao Mbogo, *Mozilla Foundation*
Jibu Elias, *Mozilla Foundation*
Joycelyn Streator, *Prairie View A&M University*
Kathy Pham, *Harvard University*
Ziyaad Bhorat, *University of Southern California*
Kathy Pham, *Harvard University*
Steven Azeka, *Columbia University*
Contact: chao@mozillafoundation.org

There is a vibrant body of work on responsible computing pedagogy across the world. However, there are comparatively fewer programs in the Global South compared to their counterparts in the Global North. This lightning talk describes our current efforts to scale the Responsible Computing Challenge (RCC) globally. The project was initiated in 2018 across the U.S. with its first iteration transforming undergraduate curricula at 17 universities in the country. In 2022 RCC was implemented in Kenya across eight universities, with a call for proposals recently launched in India. Scaling the project aims to increase the impact of responsible computing pedagogy across the world with an emphasis on developing curricula for and within local contexts. Existing inequities that are embedded within and amplified by computing technologies point to a need for a new generation of technologists who can draw from an interdisciplinary toolkit to re-imagine computational innovations. Faculty and students are invited to attend the presentation to engage in lessons drawn from integrating sociotechnical perspectives in computing courses and consider collaborating in RCC towards global impact in prioritizing responsible software and systems. In this lightning talk we will discuss the project's structure, impact examples from the pilot implementation, early lessons from expanding to Kenya and India, and ideas for collaboration in the global expansion.

Keywords: Responsible Computing; Global South; Design Frameworks

DOI: https://doi.org/10.1145/3626253.3635344

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Self-efficacy Interventions for CS1

Alex Lishinski, *University of Tennessee Knoxville*
Hanhui Bao, *University of Tennessee Knoxville*
Joshua Rosenberg, *University of Tennessee Knoxville*
Contact: alishins@utk.edu

Self-efficacy is important for success in computing courses, but best practices for how computing instructors can support students' self-efficacy development are still a matter of ongoing development. CS students' judgments about their own abilities are often very self-critical and disagree with the ways that experts in CS would measure competency. Self-assessment scaffolding has been observed to improve learning outcomes and self-efficacy in various domains. However, these practices are not well known in CS education research, and there is a lack of empirically-based guidance about how to implement them. This project is devoted to developing a self-assessment scaffolding intervention for CS1 that is effective, efficient to implement, and usable for students. Improving pedagogical practices in introductory courses (CS1) will help improve the outcomes for everyone, but approaches targeting self-efficacy may particularly benefit students from minoritized groups in CS, such as women and students of color.

Keywords: CS1; Self-efficacy; Self-regulated learning

DOI: https://doi.org/10.1145/3626253.3635348

Creative Labs in a CS1 Course Self-directed Labs Enhance Individuality in Computer Science Learning

Melinda McDaniel, *Georgia Institute of Technology*
Contact: mcdaniel@cc.gatech.edu

Most CS1 classes are designed to teach programming and computational thinking using a specific programming language such as Python or Java. This type of in-depth study leaves little room for creative exploration by students. In particular, at large colleges and universities, the programming assignments are usually autograded to keep up with expanding enrollment. This lightning talk will introduce the idea that simply adding three creative labs that do not use the actual course material, but instead expose students to other computing concepts, can give the students who are struggling a chance to catch up, while also offering them a much-needed breather doing something in computing that is individually directed and creative.

Over the last few years, my Teaching Assistants (TAs) and I have developed three simple web development labs as assignments for our large CS1 classes. These three labs are usually given around each of the exam weeks and provide a much-needed break for the students from the regular class material that continues to build upon prior learning. Many students report that the labs were their favorite part of the course because they were creative and self-directed.

This lightning talk aims to gather feedback on this concept and assess its potential for integration into advanced CS courses. I'm also keen to hear suggestions for alternative labs in a CS1 class that would both introduce students to further CS principles and allow them to showcase their creativity.

Keywords: CS1; Creativity; Computing Education; Individualized Instruction

DOI: https://doi.org/10.1145/3626253.3635342

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Moms can be Computing Leaders, Too!: Why We need Computing Community Learning Centers Designed and Lead by Mothers

Patricia Ordóñez, *University of Maryland, Baltimore County*
Contact: patti.ordonez@umbc.edu

Women in underserved communities of computing such as women of color, women in poverty, elderly women, women with disabilities, and mothers who prioritize their families over their careers have little to no access to training or career development in a technical field due to lack of childcare, limited availability, lack of education, lack of employment history, transportation, and financial cost of training. Women from these groups often suffer from trauma or imposter syndrome lacking the confidence to believe they are capable of working with technology. Yet, in 2022, women are 91% of the households with children in public housing, which is more than 232 thousand households. Children from these households have very little possibility of social mobility from the bottom to the top quintile. There is a dearth of computer training opportunities for these women. A community computing learning center that is inclusive of and tailored to all mothers and their children would help to fill the void in the US technical workforce and break the cycle of poverty that exists in many low-income communities. Universities can play a critical role in the participatory design of these centers so that they can dually serve as an equity-enabling computer education research center for its faculty and students. Women led the computing revolution in the past and can lead the culturally responsive computing education movement of the future.

Keywords: women, women of color, elderly women, poor women, women with disabilities, mother, computing education, equity-enabling research, culturally responsive education, informal learning, nontraditional pathways, community-based learning, participatory design, social mobility

DOI: https://doi.org/10.1145/3626253.3635353

Data Analytics for Social Good: A Collaborative Fusion of Computer Science and Social Science

Tina J. Ostrander, *Green River College*
Tim Scharks, *Green River College*
Kendrick Hang, *Green River College*
Contact: tostrander@greenriver.edu

CS 109, "Data Analytics for Social Good," exemplifies interdisciplinary education, preparing students for a data-centric world where technology and social consciousness unite. The course is co-taught by a Computer Science (Software Development) instructor, and a Social Science (Geography) instructor. This introductory course empowers students to harness the power of data for addressing societal issues. Participants learn data collection, computation, analysis, and visualization techniques while exploring themes of equity and socioeconomic inequality. Through data-driven advocacy, students emerge as catalysts for social change and justice. No prior experience is necessary.

Our lightning talk will delve into how CS 109 breaks traditional disciplinary boundaries to achieve the following learning outcomes:

- Create Computational Artifacts: Students develop practical computing skills to tackle real-world problems.
- Collaborative Learning: Collaboration is at the core of our course, fostering teamwork in both programming and problem-solving.
- Abstract Thinking: Students master abstraction principles while writing programs or creating computational artifacts.
- Data-Driven Insight: We empower students to discern patterns, test hypotheses, and gain insights into equity and socioeconomic inequities.
- Advocacy Through Visualization: Students effectively communicate insights using visualization, notations, and precise language, championing social change.
- Descriptive Statistics: Our curriculum equips students to summarize data, uncover patterns, and address pressing social issues.
- Data Management: Students learn to clean, summarize, and visualize data related to social concerns.

Keywords: data analytics; social science; interdisciplinary; CS+X, social science; computing for social good; inclusion

DOI: https://doi.org/10.1145/3626253.3635355

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.

© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

A Cross-disciplinary Review of Introductory Undergraduate Data Science Course Content

Michael A. Posner, *Villanova University*
April Kerby-Helm, *Winona State University*
Alana Unfried, *California State University Monterey Bay*
Douglas Whitaker – *Mount Saint Vincent University*
Marjorie E. Bond – *The Pennsylvania State University*
Leyla Batakci – *Elizabethtown College*
Contact: michael.posner@villanova.edu

Data Science is one of the fastest growing fields with unmet demand from employers. Many academic institutions have taken on the task of creating programs to meet both current and future needs and demands. Data science, as a field, integrates aspects of computer science, statistics, and subject matter expertise which encourages cross-disciplinary conversations and collaboration. In this talk, we present results from a broad survey of instructors of introductory college-level data science courses for undergraduates. In addition, we explore the alignment of these findings with the recommendations of various professional organizations.

We conducted a national survey on topics covered in introductory, college-level data science courses. With responses from computer scientists, statisticians, and allied fields, these results represent a wide array of instructors of data science. The survey identifies topics commonly covered, the amount of time spent on each, common and divergent definitions of data science, and course materials used. These results will be presented.

We will then discuss the alignment of these results through a rigorous review and synthesis of recommendations from various professional organizations. These include Association for Computing Machinery's Computing Competencies for Undergraduate Data Science Curricula[1], the National Academies of Science, Engineering, and Medicine's Data Science for Undergraduates: Opportunities and Options[2], the Park City Math Institute's report Curriculum Guidelines for Undergraduate Programs in Data Science[3], and the American Statistical Association's Two-Year College Data Science Summit Final Report[4] and Curriculum Guidelines for Undergraduate Programs in Statistical Science[5]. We will also explore alignment with ABET's accreditation of data science.[6]

Keywords: Data Science; Undergraduate Curriculum

DOI: https://doi.org/10.1145/3626253.3635352

Exposing the Hidden Curriculum with a First Year Computing Seminar

Derek Riley, *Milwaukee School of Engineering*
Katherine Panciera, *Milwaukee School of Engineering*
Contact: riley@msoe.edu

Representation and persistence to graduation for first year students in a computing major is stubbornly disproportionate depending on gender and ethnic background. While this subject has been extensively studied, progress toward changing the overall national graduation ratios has been minimal. Addressing the first-term experience for students has promise to enable progress toward improving the disproportionality.

Self-efficacy has been shown to be an important factor for completing the first programming course at a university. Confidence erosion, lack of peer study groups, and other factors are documented sources of attrition in computing majors. First year seminars are a documented high-impact practice for students that can promote self-efficacy.

In this work we present an effort to introduce a first-year seminar course for all computing majors at the Milwaukee School of Engineering University designed to expose the hidden curriculum specific for computing majors. The seminar is an optional weekly meeting for 50 minutes that gathers first-year students in one place to discuss topics like: building a problem-solving peer network, how to successfully answer software development exam questions, who and how to ask for help, history of computing, how to manage time, and more. We introduce collaborative, motivating examples that expose students to topics relevant to their major but not explicitly taught such as coding exam strategies, regular expressions, and productivity tools. In this session, we will present our seminar structure as well as preliminary data gathered regarding it efficacy toward addressing representation and persistence gaps.

Keywords: Computer Science Education; Seminar; Hidden Curriculum; Retention

DOI: https://doi.org/10.1145/3626253.3635347

Enabling Widespread Engagement in DS and AI: The Generation AI Curriculum Initiative for Community Colleges

Rebecca Schroeder, *The University of Texas at San Antonio*
Jianwei Niu, *The University of Texas at San Antonio*
Ashwin Malshe, *The University of Texas at San Antonio*
Sue Hum, *The University of Texas at San Antonio*
Siobhan Fleming, *The University of Texas at San Antonio*
Ian Thacker, *The University of Texas at San Antonio*
Contact: rebecca.schroeder@utsa.edu

The proposed initiative aims to promote broader engagement in data science and artificial intelligence by encouraging the integration of a research-based Generation AI (GenAI) curriculum within community colleges. The GenAI curriculum encompasses interdisciplinary modules, data sets, and educational content relevant to data science, computer science, and artificial intelligence. Community colleges, being vital conduits to a substantial student demographic (as evidenced by 40% of first-time college freshmen commencing their post-secondary education at these institutions), present an opportune environment for enhancing student diversity and, consequently, diversifying the workforce in data science, computer science, and AI.

The GenAI team at The University of Texas at San Antonio endeavor to develop and implement instructor training and curriculum development workshops tailored for community college faculty. Through collaboration, GenAI and community college faculty will establish faculty learning communities, fostering the creation of novel modules and distinctive instructional materials that will seamlessly integrate into the established computer science and data science curricula. At this nascent stage of the project, our team is eager to receive valuable input regarding potential theoretical frameworks, exploration of pertinent existing approaches, and expressions of interest for collaboration, all of which are pivotal in shaping the design and successful implementation of this initiative.

Keywords: Artificial Intelligence; Data Science; Higher Education; Underrepresented Minorities; Community Colleges; Experiential Learning

DOI: https://doi.org/10.1145/3626253.3635343

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Keeping Software Engineering Curriculum Relevant

Simon Sultana, *Reedley College*
James D. Kiper, *Miami University*
Brent Auernheimer, *California State University, Fresno*
Gursimran Singh Walia, *Augusta University*
Contact: simon.sultana@reedleycollege.edu

This lightning talk focuses on the efforts of software engineering (SE) academics to keep curriculum relevant. SE evolves rapidly, with changing technology and industry expectations. The curriculum review bodies (e.g. ACM and IEEE-CS working groups) provide a foundation for curriculum, but can have refresh cycles measured in years. We aim to identify, and potentially increase, the confidence level of SE educators with respect to the relevance of their curriculum. It is also of interest to identify the sources academia uses to keep their courses current. and the role of gray literature (GL). Other fields have found GL useful in bridging academic research and industry needs. GL can be extended to SE to aid faculty preparing students for industry.

We address two questions: first, given the velocity of technical change, do SE educators feel their courses effectively prepare students for their careers, specifically with regard to recent developments? Do current curricular guidelines accurately reflect industry practice and needs for our graduates? Second, how do we track current and emerging trends to capture relevant competencies? A study of scholarly literature will have a limited impact on our understanding of current and emerging trends. Instructors would do well to utilize other sources to supplement in topic identification and mastery level. The role of GL in this process is of particular interest. It is hoped this discussion can help guide a study which would investigate the development of a best practice model for maintaining relevant SE curriculum.

Keywords: software engineering, curriculum, gray literature

DOI: https://doi.org/10.1145/3626253.3635346

DEEILS: Data Ethics Embedded Interactive Learning System for Computer Science Students

Ke Yang, *University of Texas at San Antonio*
Contact: ke.yang@utsa.edu

In the rapidly evolving computing technology landscape, ethical issues arise at every stage of the data lifecycle, from collection to downstream predictive analytics, including but not limited to pre-existing bias, privacy, fairness, and accountability of machine learning and artificial intelligence algorithms. Data ethics education for students in Computer Science and related STEM programs has become a focal point of discussion and innovation. This work introduces a system, DEEILS, that allows students to learn about ethical issues at each stage of the data lifecycle through multi-media interactive modules and real-world scenario simulations. It also supports instructors with customized components for different levels of courses. DEEILS consists of three phases: *Collection*, *Preparation*, and *Analytics*, which represent the stages of how data is created and evolves through its lifecycle. Within each module, DEEILS integrates components to simulate the steps that data practitioners take in real-world scenarios, such as cleaning, de-identification, and feature engineering in the *Preparation* phase. It then guides students through the ethical issues that can arise in each component, using examples of real datasets and commonly used computational techniques for that component. Through interactive content with real-world applications, DEEILS aims to provide an adaptive, immersive learning environment to facilitate education on ethical issues in data-driven science.

Keywords: data ethics**;** interactive learning**;** Computer Science students

DOI: https://doi.org/10.1145/3626253.3635357

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

The Role of Probing and Clarifying Questions for Teaching Fellows in Computer Science: Guiding Student Growth

Yuliia Zhukovets, *Harvard University*
Carter Zenke, *Harvard University*
David J. Malan, *Harvard University*
Contact: yuliia@cs50.harvard.edu

As CS enrollments have grown and professors have increased their reliance upon undergraduate teaching assistants (TAs), it has become all the more important to identify strategies that equip TAs to assist students with coursework while providing the best learning experience during teaching sessions. In this lightning talk, we present a strategy consisting of clarifying and probing questions that highlight the importance of equipping students with tools to solve computer science problems on their own, instead of seeking help from the teaching staff every time they face a challenge. Toward this end, we will share a guide that we adopted during TA training for Harvard University's introductory course, CS50. Additionally, we will share scenarios in which such strategies might be most useful, as during group tutorials or one-on-one office hours. Lastly, we will share preliminary findings on how the strategies can not only benefit a student during the time when a TA is helping them but also equip them with skills to solve such problems on their own.

Keywords: introductory computer science; teaching strategies; undergraduate teaching assistants; teaching assistants training; probing and clarifying questions.

DOI: https://doi.org/10.1145/3626253.3635358

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

SIGCSE TS 2024 Nifty Assignments

Alphabear Partial Solver

Ryan E. Dougherty, *United States Military Academy*
Contact: ryan.dougherty@westpoint.edu

Programming a solver for a video game is a simple way for students to become engaged with solving complex problems within CS1 courses. This assignment involves writing a partial solver for the Alphabear video game, which is to find English words among letters present on a board with decreasing timers. The goal of the game is to maximize earned points, and using each letter before its timer runs out achieves that goal. This partial solver is to determine the "best" word choice at a given point in the game. Students here are to assign a weight to each word, which is specified in the assignment, and to find any word of smallest weight. Some skills necessary to solve the assignment are reading potentially large files, designing a nontrivial algorithm to calculate which word is best, and being exposed to different weighting functions.

Keywords: intro c++, cs1, nifty assignment

DOI: https://doi.org/10.1145/3626253.3635331

Exploring the Scurry of Squirrels in Central Park

Rachel Lyn-Salmon Frisbie, *Michigan State University*
Devin Silvia, *Michigan State University*
Marcos D. Caballero, *Michigan State University*
Rachel Roca, *Michigan State University*
Amanda Bowerman, *Michigan State University*
Krithi Sachithanand, *Michigan State University*
Contact: salmonra@msu.edu

As computing becomes increasingly intertwined with other disciplines, research that centers computing education in an interdisciplinary context (and the challenges surrounding it) is increasingly relevant. In particular, writing modular code (i.e. using functions) is a fundamental part of scientific programming. However, functions have been identified as challenging for students to learn. This assignment leverages a fun, real, and approachable dataset from the 2018 Central Park Squirrel Census as well as experiences authentic to developing scientific programs to introduce the concept of functions in Python. The assignment is intended to be delivered in a "flipped classroom" format, where students are first introduced to concepts in videos and short problems prior to coming to class. Once in class, the students work collaboratively in groups of 4-6 with a hands-on programming activity. By the end of the assignment, students have not only had the opportunity to learn about functions in an authentic context, but they have also been able to make calculations and propose their own problems to learn about the data.

Keywords: computing education; Python; functions; interdisciplinary

DOI: http://doi.org/10.1145/3626253.3635334

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Simulating Election Votes

Aleksandr Popov, *National Research University Higher School of Economics*
Contact: alexpoov99@gmail.com

CS0 and CS1 courses are becoming an essential part of many social science programs, including sociology and political science. This assignment was designed as part of the 'Basics of Programming in Python' course for undergraduate political science students. The course aims not only to prepare students for the unified programming exam, but also to apply concepts of computational thinking in the context of political science problems. By linking these fields, students learn to overcome inner barriers in coding and explore the potential of computational social science methods for both academic path and career.

This project seeks such a connection between subjects of political science major as Comparative Electoral Systems and Quantitative Methods of Political Research, and concepts of computer science. Students process the generated votes, applying critical thinking to identify spoilt ballots, and then formulate rules for counting votes (First-past-the-post, Borda count, Condorcet winner criterion) in algorithmic form. Optionally, a short essay can be added in which students are asked to speculate about political leanings, the transitivity of preferences and the advantages/disadvantages of electoral systems based on analyses of the data in the assignment. As the result, students not only develop hard skills in programming, but most importantly conceptualize complicated topics through the prism of their specialization, fostering the ability to see the connections between these fields and understand how they can work together to address complex real-world scenarios.

Keywords: computational social science, teaching computational thinking

DOI: https://doi.org/10.1145/3626253.3635336

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Infinity War

Yashas Ravi, *Rutgers University – New Brunswick*
Ana Paula Centeno, *Rutgers University – New Brunswick*
Contact: yr271@scarletmail.rutgers.edu

The Infinity War assignment tests the comprehension of the graph data structure in a context based on Avengers: Infinity War. The assignment has various tasks, where each task focuses on a specific aspect of a graph, and makes connections to the movie as well as other fields such as Physics, Neuroscience, and Philosophy in a creative manner. The assignment tests graph manipulation, connectedness, search, recursion, cycles, and other topics, where each task focuses on one of these topics. Furthermore, the assignment is meant for CS2 students, takes around 1-2 weeks to complete, and has a mix of easier and harder tasks. The student handout, starter code, and a website containing the metadata of the assignment are provided.

Keywords: Data Structures; Graphs; Shortest path

DOI: https://doi.org/10.1145/3626253.3635332

The Fingerprint Assignment: An Interdisciplinary Assessment for CS I Education

Christian Servin, *El Paso Community College*
Ivan Alonso, *El Paso Community College*
Emiliano Garcia, *El Paso Community College*
Contact: cservin1@epcc.edu

The fingerprint assignment is tailored for the Fundamentals of Computer Programming I and II courses, a.k.a. CS1 and CS2, with a primary focus on teaching the principles of two-dimensional arrays and object-oriented using Java. While these concepts are core to CS1, the assignment is divided into three sections that delve into introductory computing topics, emphasizing cybersecurity and data analytics to captivate students' interest in CS1. This approach is driven by industry recommendations and education agencies. The assignment has been effectively implemented in both CS1 and CS2 classes within a two-year computer science program offered at an HSI community college. It serves a diverse range of audiences, including traditional CS1 and CS2 assessments, acting as a remedial tool for CS2, a project for CS1 honors students, and a summative assessment for Competency-Based Education (CBE). The assignment aligns with specific Student Learning Outcomes outlined in three distinct ACM curricular guidelines, including the ACM/IEEE-CS/AAAI CS2023. Additionally, it leverages "Bloom's for Computing: Enhancing Bloom's Revised Taxonomy with Verbs for Computing Disciplines" to stimulate various levels of cognitive understanding throughout the assignment. Furthermore, it consolidates topics typically covered in CS1, synthesizing these concepts into a single, comprehensive activity. This assignment also recognizes Competency-Based Education (CBE) as a formative assessment method suitable for a wide range of technical and non-technical programs. Lastly, it offers a flexible delivery structure, allowing educators to seamlessly integrate it into their classrooms in a modular fashion.

Keywords: CS1/CS2; Curricula Guideline; community colleges; two-year programs; CS2023; data sciences; cybersecurity

DOI: https://doi.org/10.1145/3626253.3635333

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.
© 2024 Copyright is held by the owner/author(s)
ACM ISBN 979-8-4007-0424-6/24/03.

Modular Virtual 3D Cities Assignment

Christopher John Tralie, *Ursinus College*
Contact: ctralie@alumni.princeton.edu

The purpose of this intro to CS assignment is to give students practice with modularity and method design in a fun application: designing virtual 3D city environments. There are two versions of the assignment provided: one in C++ and one in Java. Both write HTML code to a file which renders the scene in a browser using three.js. Visual Studio code with a live web server extension makes it seamless to develop and update the 3D scenes interactively. Students start with various overloaded methods for drawing primitive 3D shapes (e.g. cylinders, ellipsoids, rectangular prisms) in different colors and proportions. Students then put these shapes together in methods that draw different city objects (e.g. tree, fire hydrant, stop light, sedan). In the process of making these methods, students practice reading method documentation of the primitive shapes, as well as writing their own method documentation for the method parameters specifying positions and orientations of each city object. Once the individual city objects are completed, students go up in abstraction to create a method that assembles a city block consisting of all of the objects together. They then finish the assignment with a method that calls the city block method in a loop to create an entire city consisting of multiple city blocks. After creating a basic city, students submit a creative virtual environment to a class gallery. A full description, along with amazing past student art contest submissions, can be found at http://www.ctralie.com/VirtualCities/

Keywords: Nifty Assignments; Scene Graphs; Computer Graphics; three.js; Modularity

DOI: https://doi.org/10.1145/3626253.3635335

Author Index

Abbasalizadeh, Maryam 1105
Abel, Philip 39
Abichandani, Pramod 1546
Acharya, Subrata 1895
Adams, Joel 4
Adegbile, Atinuke 39
Adeleke, Michael B. 771
Adepu, Manas 18
Adjei, Seth 1882
Adler, Rachel F. 680, 687
Adrion, W. Richards 1548
Afuye, Lekan 39
Agarwal, Arav 1550
Agbo, Friday Joseph 11
Aggarwal, Ashish 18, 1883, 1887
Ahluwalia, Dev 1568
Ahmed, Ahmed 736
Ahmed, Sadaf 32
Ahmed, Zeyad 1877
Ai, Xinyi 1230
Aivaloglou, Efthimia 262
Akgun, Mahir 25, 1552
Akhmetov, Ildar 32
Akhtar, Saima 1154
Akolekar, Harshal D. 625
Akram, Bita 526, 1521, 1680, 1762
Akullian, Jennifer 597
Alabi, Daniel 39
Alaofi, Suad 46
Alawini, Abdussalam 359
Albert, Jennifer 1554
Aldirawi, Hani 1934
Alex, Sharon 1668
Al-Hossami, Erfan 53
Ali, Mohd Hasan 1435
Alipour, Amin 67
Allan, Vicki 958
Allee, Willliam 1286
Allen, Brittney 1746
Allison, Zelda 826
Alm, Cecilia O. 1556
Almogbil, Atheer 1696
AlOmar, Eman Abdullah 60, 1923, 1930
Alonso, Ivan 1943
Alterovitz, Ron 1676
Alvarado, Christine 604, 1119
Amarouche, Thezyrie 296, 1670
Amoozadeh, Matin 67
An, Marshall 116, 1826
Anderson, Hayworth 4
Andrei, Oana 81
Angeli, Lorenzo 1728, 1923
Antoine, Allen 1602
Anu, Vaibhav 1630
Anvik, John 1836
Arce, Candy 1877
Arriola, Sheryl 1788
Arunachalam, Nimmi 74
Aryal, Manogya 1882
Ashburn, Kyle 1000
Ataullah, Mohammad Osama 625
Atchison, Kate 694, 1658
Aucie, Andrew 1670
Auernheimer, Brent 1939
Avery, Charlotte 1558
Awoyemi, Idowu David 764
Ayuno, Kezziah 32
Azeka, Steven 1941
Babalola, Damilola 526, 1646
Bacher, John 1818
Badger, Kelsey 1764
Badrinath, Anirudhan 1896
Bae, Haesol 1710
Bagui, Sikha 1588
Bailey, Reynold 1556
Bailey Lee, Cynthia 1869
BaileyShea, Chelsea 1556
Baker, Catherine M. 324, 1892
Baker, Ryan S. 214
Baker, Stephanie N. 1409, 1602, 1910
Ball, Michael 1896, 1897, 1924
Balladares, Fernando 498
Ballou, Elijah 771
Banson, Justice 757
Bao, Hanhui 1935
Barapatre, Deepali 874
Barboza, Luiz 1892
Barker, Lecia 269, 470, 1636
Barkhuff, Grace 1560, 1562, 1720, 1917, 1921
Barnes, Tiffany 1402, 1456, 1792, 1830, 1904
Barnet, Mike 1929
Barr, Matthew 81
Barr, Valerie 1908
Barrera Machuca, Mayra D. 1808
Barrett, Joanne 1566
Barros, João Paulo 1564
Bart, Austin Cory 1636, 1893
Bartucz, John 1865
Basit, Nada 1684
Bastian, Nathaniel D. 248
Basu, Debarati 1758
Basu, Satabdi 1804
Batakci, Leyla 1702, 1937
Bates, Diane C. 1782
Bates, Rebecca 1922
Batra, Anna 88
Bauer, Christina 1237
Baumer, Sven 590
Bausch, Gillian 95
Bean, Nathan H. 190, 1910, 1928
Becker, Brett A. . 296, 799, 1063, 1916
Bellettini, Carlo 102
Beltran, Emiliano 1428
Benedicto, Dana 1810, 1834
Bennett, Andrew 1566, 1863
Berg, Roseann 456
Berg, Sam 632, 826
Bergner, Nadine 1279
Bernard, Connor Robert 1568
Berner, Maverick 1570
Berner, Maximus 1570
Bevilacqua, Joey 1572
Bezáková, Ivona 1584
Bhalerao, Rasika 1574
Bhalli, Noshaba 1576
Bhaskar, Niharika 109
Bhattarai, Shreeansha 1670
Bhorat, Ziyaad 1935
Bhutada, Rhea 1926
Bhuyan, Jay N. 547
Bigonah, Maryam 1694
Billingsley, William 1161
Blair, Jean R.S. 248
Blank, Adam 1869

Blanton, Melanie 1554
Blaser, Brianna 597, 1286, 1915
Bobbadi, Mahesh 1402
Bogart, Christopher .. 116, 1826, 1842
Bohannan, Madison 1810, 1834
Boisvert, Deborah 972
Bolante, Kianna 123
Bolger, Emily 1930
Bolte, Mario 443
Bond, Marjorie E. 1702, 1937
Bond, Sally 694
Bopp, Chris 130
Boreczky, Jack 1853
Borenstein, Jason 1560
Borrelli, Thomas J. 137
Bosch-Wilson, Kathleen L. 1768
Bourgeois, Anu G. 1716, 1748
Bowdon, Jill 1126
Bowerman, Amanda 1941
Boyer, Kristy Elizabeth 639, 1272
Brankovic, Ljiljana 1161
Braught, Grant 144
Brender, Jérôme 1580
Brewster, Ashley 1774
Bricker, Lauren 1907
Brilliantova, Angelina 1584
Brock, Janet 1830
Brodley, Carla 729, 909, 1535
Brooks, Alexi 1921
Brooks, Christopher 1258
Broquedis, François 1189
Brown, Bailey 151
Brown, Matthew L. 1710
Brown, Neil C. C. 158, 165, 1931
Brown, Noelle 979, 1878
Brunskill, Emma 200, 736
Brusilovsky, Peter 1904
Brylow, Dennis 380, 1570, 1786
Budhiraja, Ritvik 625
Bui, Giang 1237, 1578
Bumbacher, Engin 1580
Bunescu, Razvan 53
Burdge, Darci 1912
Burke, Quinn 1582, 1608
Burnett, Margaret 1035
Burns, Kaylee 736
Burton, Emanuelle 1523
Bush, Jeffrey B. 1674
Butler, Zack 1584
Caballero, Marcos 1930, 1941
Camarena, Maria 1869
Cambaz, Doga 172
Cameron, Melissa 1586
Cameron, Sarah 1588
Campbell, Bradford 923
Cannady, Justin 1590
Cantrell, Katherine 1766
Caraco, Serena 179
Carbajal Juarez, Kitana 1742, 1846
Cardarelli, Joseph 1890
Carrington, Luke Rivers 1435
Carroll, Meredith 394
Carruthers, Sarah 1592
Carvalho, Marco 986
Castillo, Darian 1654
Castle, Sarah D. 1594
Castro, Francisco 1914, 1919
Castro, Lauryn M. 1766
Cateté, Veronica 1456, 1660, 1718, 1402
Cazalas, Jonathan 186
Celepkolu, Mehmet 639
Centellas-Claros, Leonardo 951
Centeno, Ana Paula 1942
Chakraburty, Srijita 639
Challen, Geoffrey 193, 1525
Chan, Albert 1893
Chapeton-Lamas, Nery 1531
Char, Bruce W. 1931
Chatterjee, Amreeta 1035
Chaturvedi, Naina 1898
Chen, Hongxuan 193, 1056, 1722
Chen, Huanyi 1596
Chen, Kevin 123
Chen, Lujie Karen 576, 1598, 1894
Chen, Quan Ze 123
Chen, Stella 67
Chen, Tianyi 1484
Chen, Xi 1600, 1851
Chen, Ying 1421
Chen, Yu 1832
Chen, Yuxin 1670
Chen Shui, Zaria 352
Cheng, Alan Y. 200
Chewar, Christa M. 248
Childs, Joshua 1602
Chiodini, Luca 1572, 1894
Chiu, Meng-Chieh 207
Choe, Jacob Seungwon 1604
Choi, Heeryung 1258
Choi, Suh Young 88
Choudhary, Arihant 1606
Christie, Aaja 1748
Chu, Sharon 778
Chu, Wilson 1568
Clark, Nicholas J. 1632
Clayville, Kristel 1523
Closser, Florentina M. 1710
Cloude, Elizabeth B. 214
Cobb, Morgan 1857
Cobo, Alexis 221, 1863
Coenraad, Merijke ... 227, 1582, 1608, 1788, 1800
Cohen, Taya 1414
Cohney, Shaanan 1175
Contreras, Gabriel 1286
Cooper, Stephen 463
Coore, Daniel 352
Cornell, Kimberly A. 345
Cornwall, Elyse 736
Corsi Ferrão, Rafael 303
Cosman, Benjamin 1056
Cota, Raena 1840, 1877
Cowit, Noah Q. 234, 1610
Cox, Katlyn 771
Coyne, Steven 533
Crabb, James 241
Crawford, Chris 764
Crick, Tom 799
Cruickshank, Iain J. 248
Cui, Jialin 255
Cunningham, Heather 1126, 1768
Cunningham, Kathryn 193
Cutter, Pamela 1612
Cutts, Quintin 1168
Czajka, Sandra 1875
Dai, Kunchen 1924
Daniels, David 67
Danukarjanto, Luki 708
Darabi, Rona 109
Das, Udayan 1614, 1616
Dasgupta, Dipankar 1435
Dastur, Delnavaz 1531
Davenport, James H. 799
Davis, Destiny 771
Davis, Richard Lee 1580
Dawson, Kelsie 1698
Dawson Head, Erika 1895
De Menego, Luca 1923
de Wit, Shirley 262

Dean, Julia 269
Dean, Victoria 1525, 1908
Decker, Adrienne 276, 819, 1732, 1915, 1932
Dee, Thomas S. 1933
Dehbozorgi, Nasrin 1618
Dekhane, Sonal 283, 1895
del Carpio Gutierrez, Andre 289
del Vado Vírseda, Rafael 1620
DeLyser, Leigh Ann 221
Delzanno, Giorgio 1622
DeMars-Smith, Julianne 1925
Denny, Paul 289, 296, 519, 1063, 1223, 1901, 1916
Denzler, Benjamin 1361, 1624
Derbinsky, Nate 1525
Dev, Harshal 625
Dey, Indrani 1674
Dhama, Sunny 639
Dimino, James 1740
Dindukuri, Sahithi Charitha 1105
Do, Ellen Yi-Luen 1930
Dodds, Zachary 1908
Doore, Stacy A. 1523, 1921
dos Santos Montagner, Igor 303
Dougherty, Ryan E. . 310, 1626, 1628, 1632, 1941
Doughty, Jacob Arthur 1550
Douglass, Esther 1630
Downey, Kelly 317
Doyle, Aidan 1550
Drozda, Zarek 1527
DuBow, Wendy M. 1529
Duhme, Christof 1293
Dunbar, Kyle 1582, 1608, 1788
gDy, Joshua Caleb 554
Dziallas, Sebastian 1028
Earle-Randell, Toni 639
Earth, Steve 1931
Eaton, Cyntrica 1463
Ebling, Maria R. 1632
Edmison, Bob 276
Edwards, Doug 1746
Edwards, John 1347
Edwards, Katlyn 1925
Edwards, Stephen H. 276, 1915
Eguchi, Amy 1926
Eiselt, Kurt 965
El Otmani, Salma 1634
Elglaly, Yasmine N. 324, 944, 1892
Elias, Jibu 1935
Elizabeth Margulieux, Lauren .. 1168
Ellis, Heidi J. C. 1780, 1912
Ellis, Margaret 1586
Englert, Megan 1636
Erete, Sheena 1098, 1638
Erez, Yael 1899
Ericson, Barbara J.... 1000, 1662, 1686
Erwig, Martin 1007
Esche, Svana 331
Espinoza II, Douglas Manuel 1435
Etim-Andy, Miracle 1638
Everson, Jayne 673, 1917
Ewoldt, Kathy B. 1766
Exter, Marisa E. 1896
Fan, Aysa X. 1640
Fancsali, Cheri 1816
Fang, Joana 109
Farghally, Mohammed 338
Fariha, Anna 1258
Farley, Emy 1084
Farrell, Andrew 1740
Feijóo-García, Pedro Guillermo 1932
Feldhausen, Russell 1907, 1910, 1928
Feliciano, Bernado 95
Feng, Haoran 519
Fereidooni, Sonia 88
Ferguson, Camille 1814
Ferguson Bey, Brian 701
Fernandez, Amanda S. 345, 1428, 1642
Ferreira Mello, Rafael 1892
Ferrer, Ray 1814
Ficke, Cherrise 394
Fields, Deborah A. 874, 1533
Fierro, Gabriel 1794
Fiesler, Casey 234, 1610, 1921
Finn, Chelsea 736
Fisk, Susan Rebecca 1456
Fisler, Kathi 1133, 1527
Fitzpatrick, KellyAnn 1918
Flatland, Robin 1855
Flemming, Siobhan 1938
Fletcher, Carol 1602
Flores, Madelen 1568
Flores, Pablo 498
Floryan, Mark 1684
Foerst, Anne 130
Fokum, Daniel T. 352
Fong, Morgan M. 359, 1652, 1917
Forden, Jack 380
Fossum, Timothy 1900
Fouh, Eric 214
Fowler, Max 562, 1652
Fox, Alexander 1879
Fox, Armando 179, 1604, 1810, 1834, 1853, 1909
Franklin, Diana 1335, 1722
Fraser, Gordon 401, 408
Freeman, Charity 1909
Freeman, Jason 1746
Friend, Michelle 1907, 1910
Fries, Mary 1897
Frisbie, Rachel Lyn-Salmon 1941
Fu, Joyce 1934
Fukazawa, Yoshiaki 569, 1838
Fulton, Chris 1616
Furqan, Zeeshan 186
Gago-Masague, Sergio 1742, 1846
Galal, Abdalaziz 477
Ganelin, Daniela 1933
Ganesan, Sumukhi 1644
Gao, Chongjiu 88
Gao, Chunming 1933
Gao, Xiaopeng 1484
Gao, Zhikai 1251, 1646
Garbrecht, Lisa 1910
Garcia, Dan .....
Garcia, Daniel 1531, 1568, 1604, 1606, 1738, 1896, 1897, 1909, 1931
Garcia, Emiliano 1943
Garcia, Leiny 1648
Garcia, Rita 366
Garcia, Rosalinda 1035
Garcia, Yuan 1908, 1924
Garcia-Marin, Alex 18
Gardner-Mccune, Christina . 18, 778, 1885
Garfus-Knowles, Dylana 1537
Garrett, Stephen 1746
Garvin, Megean 151
Gautam, Aakash 373, 1914
Gaweda, Adam 1646
Gawenda, Julie V. 1650
Gebhard, Alexander 380
Gebremedhin, Assefaw 241
Gehringer, Edward 255
Geng, Tieming 1893
George, Kari L. 1652
German, Dan-Adrian 1897

Gertner, Yael ...562, 1056, 1652, 1722
Ghafoor, Sheikh 1906
Gharadaghi, Raha 1670
Ghena, Branden 923
Ghosh, Ahana 387
Ghosh, Anuradha 1848
Ghosh, Srija 792, 1740
Gilbert, Nina 1760
Gilbert, Sarah 722
Gill, Catherine 729
Giordano, Megan 909
Glantz, Sean 1907
Glass, Cheyne 1678
Glazewski, Krista D. 639, 1710
Glazewski, Zackary 1119
Glebova, Olga 1654, 1911
Glynn, Colin 1656
Goldsmith, Judy 1922
Goldwasser, Talia 151
Goldweber, Michael 1523
Goldweber, Mikey 1706, 1914
Gomes, Anabela 1244
Gomez, Kimberley 1848
Gong, Jiaqi 1894
Gong, Tianchou 1644
Gonzalez, Elias 1658
Gonzalez, Horacio 1840
Gonzalez, Luis A. 951
Gosha, Kinnis 1760
Goslen, Alex 1660
Gough, Curtice 394
Govinda Rajan, Viswajith 923
Grant, Christan 1857
Graßl, Isabella 401
Gray, Jeff 1300
Gray, Kaitlynn 1049
Greenlaw Rollins, Wyeth T. 1889
Gregg, Chris 583, 1911
Greifenstein, Luisa 401, 408
Griswold, William G. 415, 512
Grossman, Sabrina 1746
Grover, Shuchi 422, 1533
Groves, Christopher L. 1147
Gu, Xingjian 1662
Guerrieri, Benjamin Thomas 1879
Guerrini, Giovanna 1622
Guitierrez-Godoy, Clarissa 604
Gulati, Rishi 429
Gulwani, Sumit 1258
Gunawardena, Ananda 1898
Gupta, Anisha 1660
Gupta, Avi 1704
Gusukuma, Luke 1893
Gutierrez, Francisco J. 1077
Guttmann, Michael 436
Guzdial, Mark .158, 1535, 1898, 1908
Ha, Jesse 1848
Haber, Nick 1844
Hagiwara, Sumi 1630
Haji Amin Shirazi, Shirin .. 450, 1664
Hamandi, Lama 1899
Hamerski, Patti C. 1666
Hang, Kendrick 456, 1937
Hardin, Caroline D. 757
Harding, David 1576
Harper, Colton 463
Harrell, Avery 470
Harrington, Brian 477, 1668, 1670
Hartl, Anna 484
Hauswirth, Matthias 1572, 1894
Hawthorne, Elizabeth K. 1196
Haynes, Madeline 1409
Haynes-Magyar, Carl 491
Hazzan, Orit 1899
Heckman, Sarah 1112, 1251, 1646, 1871
Hed, Emily 1656
Heeren, Cinda 1909
Hegeman-Davis, Raya 562
Heiner, Cecily 1672
Helic, Denis 436
Heliotis, James 1900
Hellas, Arto ...1063, 1223, 1901, 1916
Hendrawan, Rully A. 1640
Henley, Austin Z. 1258
Henrique, Brendan 1880
Herbert, Katherine G. 1630
Herman, Geoffrey L. 359, 1056
Hermans, Felienne 262
Hermans, Kimberly 1531
Hernandez, Angie 1877
Hernandez, Manuel 1869
Herrera, Jesus 1934
Herskovic, Valeria 498
Hesse, Robert 1656
Heuer, Ute 401, 408
Hillberg, Hannah Miller 1147
Hilliard, Monica 694
Hilton, Michael 67, 1414
Himbeault, Lauren 505
Hingle, Ashish 1880
Hira, Avneet 1929
Hislop, Gregory W. 1780, 1912
Hmelo-Silver, Cindy 639
Hoang, Nga 1674
Hoelscher, Janine 1676
Hoffman, Alex 1286
Hogan, Emma 512
Holland-Minkley, Amanda 1541
Holmes, Andrew 750, 1928
Holzberger, Doris 484
Hong, Victoria 1678
Hooper, Steffan 519
Hooshangi, Sara 1929
Hoq, Muntasir 526, 1680, 1762
Horn, Matthew 1654
Horton, Diane 533
Hott, John R. 1682, 1684
Hou, Xinying 1686
Hou, Yunfei 1934
Houchins, Jennifer K. 1688, 1690, 1806
Howard-Sarin, Brij 1881
Hsia, Justin 1924
Hu, Anne Drew 1914, 1934
Hu, Helen H. 1911
Huang, Kaiyi 1670
Huang, Shan 359
Huang, Sicong 708
Hudachek-Buswell, Mary 1917
Hug, Josh 1896
Hug, Sarah 373
Hum, Sue 1938
Hummel, Joe 1112
Hundhausen, Christopher 241
Hung, Eping E. 540
Hung, Shou-Yi 708
Hur, Jung Won 547
Hurley, Ethan 1881
Hüsing, Sven 443
Iaboni, Craig 1546
Ianni, Eric 1802
Iannini, Julianna 1882
Ibrahim, Omar 1913
Ikeda, Bryce 1676
Ingram, Vincent 771
Inventado, Paul Salvador 554
Irankhah, Elyas 1740
Isenegger, Kathleen 562, 1652
Ishizue, Ryosuke 569

Israel, Maya .. 1272, 1566, 1863, 1912
Isvik, Amy 1402
Jackson, Stoney 144, 1900
Jain, Garima 792, 1740
Jain, Rishi 1752
Jalote, Pankaj 1704
Jamal, Rifa 1692
James, Darius Ellis 221
Jamshidi, Fatemeh 1694
Janeja, Vandana P. 576, 1576
Jefferson, Thomas 583
Jetzinger, Franz 590
Ji, Jie 597
Jiang, Janet 1634
Jiang, Jin 1670
Jiang, Jinya 604
Jiang, Xiaoyi 1293
Jiao, Shuyin 1832
Jimenez, Yerika 778
Jiménez, Priscilla 1905
Jin, Mingxian 1893
Jin, Wei 283
Jocius, Robin 1554
Johnson, Jeremy 1931
Jois, Tushar M. 611, 1696
Jonassen, Lorraine 283
Jones, Abigail 1882
Jones, Stephanie 1529
Jordan, Mollie 618
Joshi, Deepti 1554
Joshi, Ishika 625
Joswick, Candace 1554
Juavinett, Ashley L. 1491, 1498
Kabbes, Joseph 1893
Kadia, Jahnvi 625
Kafai, Yasmin B. 874, 881, 1533
Kafle, Richa 604
Kägi, Alain 1889
Kaiser, Alice 1814
Kao, Yvonne 632, 826, 1688, 1690
Kapoor, Amanpreet 1885
Karakaš, Aleksandar 436
Karlstad, Audrey 826
Kastelein, Kate 1698
Katuka, Gloria Ashiya 639, 1272
Kaval, Stella 1700
Kazerouni, Ayaan M. 646
Kellogg, Brian 130
Kempa, Edward 1883
Kerby-Helm, April . 1702, 1920, 1937
Kerney, William 1913
Kerslake, Chris 653
Keylor, Eric 116
Khoo, Yi Xuan 576
Khuu, Wyman 1537
Kiesler, Natalie 1916
Killen, Heather 1335
Kim, Joseph 659
Kim, Keunjae 1710
Kingsley, Sara 1914
Kiper, James D. 1939
Kirabo, Lynn 1908
Kirdani-Ryan, Mara 1913, 1919
Kirkpatrick, Michael S. 1523, 1922
Kirova, Vassilka D. 666
Kivuva, F. Megumi 673, 1919
Kletenik, Devorah 680, 687
Klimczak, Korena S. 1216
Klokmose, Clemens Nylandsted 902
Kluthe, Timothy 1286
Ko, Amy J. 673, 1712, 1914, 1915, 1919
Ko, Shao-Heng 1634
Kobayashi, Hiroshi 1838
Koedinger, Ken 1904
Koehler, Adam T. 1650
Kölling, Michael 165
Kostick, Logan 1696
Kotz, David 611
Kramarczuk, Kristina 694
Krause, Gladys 540
Krause-Levy, Sophia 1908
Krishnamoorthy Srinivasan, Shyama Sastha 1704
Krishnamurthi, Shriram 1133
Krueger, Kori 1414
Ku, Cyril S. 666
Kučera, Luděk 1925
Kuemmel, Andrew 1875
Kulkarni, Abhishek 778
Kulkarni, Anagha 373
Kulp, Jessica T. 1764
Kumar, Aayush 67
Kumar, Amruth N. 1867
Kumar, Dhruv 625
Kumud, Vedant 1752
Kunuku, Mourya T. 1618
Kurauchi, Andrew 303
Kurkovsky, Stan 1706, 1914
Kurmas, Zachary 1708
Kushalnagar, Raja 1915
Kwak, Daehan 1770
Kwon, Kyungbin 1710
Labonte, Don 1814
Lachney, Michael 701, 1914
Ladner, Richard E. 1286, 1915
Lagesse, Brent 1049
Lais, Lennart C. 1321
Landesman, Rotem 1712
Lapan, Julia C. 715
Laracy, Joseph R. 666
Largent, David L. 1900, 1915
Lash, Todd 1909
Latulipe, Celine 505
Laufenberg, Oliver 380
Law, Catherine 1912
Lawrence, Ramon 1388
Lay, Ryan 1926
Lee, Celeste 151
Lee, Claire Seungeun 1105
Lee, Crystal 1935
Lee, Cynthia 1119
Lee, En-Shiun Annie 708
Lee, Hyejeong 639
Lee, Jeongkyu 1644
Lee, Key 1925
Lee, Leon 1668
Lee, Matthew G. 1604
Lee, Taneisha 1746
Lee Solano, Lorenzo 1714
Legg, Phil 847
Leggett-Robinson, Pamela 1042
Lehman, Kathleen J. 715, 1535
Lehr, Christa 604
Lehr, Jane 373, 646
Leinonen, Juho 296, 526, 1063, 1223, 1521, 1901, 1916
Lemmer, Bridina 1812
Lenhart, Anna 722
Lentz, Sidney 470
Leo John, Rogers Jeffrey 1230
Leonhardt, Thiemo 1279
Lester, James 1680
Levin, Michael E. 1216
Levine, Stacey 1716
Levitt, Diane 1537, 1543
Lewis, Amari N. 109
Lewis, Colleen M. 993, 1049, 1652, 1869, 1873, 1895, 1901
Leyton-Brown, Kevin 1470

Li, Ang 193
Li, Min 1021
Li, Nan 1161
Li, Qiyuan 25
Li, Ruochi 255
Li, Ruoxuan 512
Li, Yi 1883
Li, Yuanhan 1568
Liang, Gongbo 1859, 1861
Liang, Jingsai 1600, 1851
Liao, Soohyun Nam 1216
Liben-Nowell, David 1084
Lim, Hansol 1718
Lim, Jared 1720
Limke, Ally 1402, 1792
Lin, Colin 1668
Lin, Kevin 88
Lionelle, Albert 729
Lishinski, Alex 1935
Liu, Charisse 1810, 1834
Liu, Charlie 750, 1927, 1928
Liu, David 533
Liu, Evan Zheran 736
Liu, Jason 74
Liu, Jingting 109
Liu, Jonathan 1722
Liu, Mengqi 743
Liu, Rongxin ...750, 1902, 1903, 1927, 1928
Liu, Ruohan 764, 1724
Liu, Yancheng 1884
Liu, Yudong 944
Liut, Michael ...757, 1237, 1477, 1578, 1871
Lloyd, Doug 1902, 1903, 1916
Lo, Linda 1670
Lobo, Alejandro 1926
Lodi, Maliha 1670
Lojo, Nelson 179
Lonati, Violetta 102
Lopez-Delgado, Maria 1726
Lovitt, Charity 1654
Lucchetta, Jessica 1728
Lukens, Bryce 1890
Lunn, Stephanie J. 74
Luo, Feiya 764
Lupoli, Shawn 1902
Luttgen, Kim 1688
Luxton-Reilly, Andrew 289, 296, 519
Ly, Anna 757, 1730
Ly, Kevin 618
Lyman, Allie 1084
Lynch, Collin 526, 1251, 1646
M, Chethan 1258
Ma, Qianou 1640
Mac, Victoria 165
Macdonald, Randy 1907
Macdonell, Cameron 144, 1912
Mack, Naja A. 771
Mackay, Sean 1732
MacNeil, Stephen 1901, 1916
Madrigal, Alejandro 1602
Maeda, Atsuhiko 1734
Magda, David 778
Magnani, Meica 1921
Mahinpei, Romina 1736
Mahipal, Vaishali 792, 1740
Mak, Giselle 1654
Malan, David J. 750, 1902, 1903, 1916, 1927, 1928, 1940
Malave, Laura 1913
Malhotra, Vedansh 1738
Malik, Ali 785, 1442
Malmi, Lauri 1223
Malshe, Ashwin 1938
Malva, Liina 387
Manley, Eric D. 1014
Mann, Carl 394
Mansingh, Gunjan 352
Marathe, Siddharth A. 1604
Marchese, Maurizio 1923
Marghitu, Daniela 1694
Marlowe, Thomas J. 666, 1630
Marousky, Annmargareth 1788
Marques Samary, Maíra 1883
Marshall, Stefanie 1539
Martin, Fred 95, 792, 1740
Martin, Nicole D. 1409
Martinez Neda, Barbara .. 1742, 1846
Martinez Rivera, Lydia 1688
Marward, Justin 1105
Mason, Raina 799
Massachi, Talie 1744
Mathapathi, Shivakumar 1899
Matlen, Bryan 1690
Matsumoto, Stephanos 1908
Matthews, James 1855
May, Ann 1084
Mazumder, Syeda Fatema 806
Mazzone, Samuel B. 1884
Mbogo, Chao 1935
McCall, Lauren 1746
McCarron, Brigid M. 611
McClintock, Declan Andrew 812
McDaniel, Emma L. 1748
McDaniel, Melinda .1911, 1917, 1936
McGill, Monica M. 819, 1820, 1822, 1871
McIlraith, Sheila A. 533
McIntyre, Miranda 1934
McKenna, Jason 1566
McKinney, David 632, 826
McKlin, Tom 639, 1456, 1746
McSkimming, Brian M. 276
Melton, Madison 833
Melville, Elizabeth 1154
Mendes, António 1244
Mendez, Rolando 1892
Mendez Mendez, Jenny 1738
Mendoza, Marcelo 951
Meneely, Andrew 840
Menezes, Tyler 456
Menier, Amanda 151
Meyers, Benjamin S. 840
Meza, Federico 1750
M'Hiri, Faten 743
Michaeli, Tilman 484, 590
Midgett, Raven 1660
Milenkovic, Lisa 1788
Miller, Kris 317
Miller, Ph.D., Mark Leslie 1899
Mills, Alan 847
Mills, Kelly 1582, 1608
Min, Wookhee 1660, 1718
Minagar, Sepehr 853, 1161
Minkowitz, Rachel 687
Minnes, Mia 109
Mirderikvand, Foozieh 95
Miroyan, Mihran 860, 1752
Mirza, Diba 1119
Mitchell, John 1395, 1844
Mitra, Chancharik 1752
Mitra, Sayan 625
Mittal, Karthik 1550
Mkaouer, Mohamed Wiem 60, 1923, 1930
Mochetti, Karina 1754
Mock, Lauren 1896, 1897
Moeller, Babette 1816
Mohammed, Mostafa 867

Monaco, Alida 39
Monga, Mattia 102
Montalvo, Audria 1381
Montes De Haro, Camilo 673
Moosvi, Firas 1736, 1909
Morales, Flor 1742
Morales-Navarro, Luis 874, 881
Moreno Santos, Igor 1572
Morpurgo, Anna 102
Morreale, Patricia 1035, 1770
Morrison, Alistair 81
Morton, Chloe 826
Moss, Eliot 207
Mostafavi, Behrooz 1904
Mott, Bradford 1660, 1680, 1718
Moudgalya, Sukanya Kannan 888, 895
Mourning, Chad 1756
Mousa, Raneem Emad..... 1885
Mouzaoir, Samuela 701
Moya, Jesse 1855
Moz-Ruiz, Dahana 1035
Munoz-Gama, Jorge 498
Munzel, Griffin 826
Murphy, Christian 597, 659, 1912
Murray, Tracey 1654
Musaeus, Line Have 902
Musicant, Dave R. 1084
Muthukrishnan, Smrithi 1660
Muzny, Felix 909
Nabi, Syed Waqar 81
Nadeem, Azqa 916
Najjar, Nadia 1758
Nakazawa, Mario 1541
Nakvosaite, Agne 1650
Nalluri, Rohita 477, 1477, 1670
Nam, Daye 67
Namukasa, Maureen 394
Narain, Sashank 1105, 1740
Narasimhan, Giri 74
Nasiha, Faiza 477
Nasir, Nabeel 923
Nasrin, Fatema 764
Neely, Leslie Cockerill..... 1766
Negron, Anthony 1814
Nelson, Brooke 1885
Nelson, Connor 930, 937
Nelson, Dana 1672
Nelson, Whitney 1760
Nelson-Fromm, Tamara ..1908, 1917
Nevarez, Erick 1840
Newby, Nigel 1886
Newman, Pax 944
Neyem, Andres 951
Nguyen, Ha 958
Nhien, Chantra 1182
Ni, Lijun 95
Nie, Allen 736
Niess, Margaret 1007
Nikolopoulos, Dimitrios 1586
Niosuha, Rose 1762
Nitta, Christopher 965
Niu, Jianwei 1938
Nnamani, Manee Ngozi M..... 972
Norouzi, Narges 860, 1521, 1604, 1680, 1752, 1762, 1810, 1834
Notess, Susan 1929
Nurollahian, Sara 979
Nurue, Herart Dominggus..... 1886
Nutwell, Emily 1764
Oconnor, TJ 394, 986
Ogden, Kaitlin 1602
Oguibe, God'Salvation F..... 1766
Oh, Diana Dajung 1768
Ojeda, Daniel 1770
Ojeda-Ramirez, Santiago 1648
Ojha, Vidushi 234, 993, 1049, 1652, 1908
Okyere-Badoo, Joel 1881
Oliveira, Eduardo 1175
Olney, Andrew M. 1892
Omidiya, Fawaz 1670
Omojokun, Fisayo 1917
Omojokun, Olufisayo 1918
O'Neill, Abigail 1700
Opdahl, Tyanin 944
Orban, Chris 1772
Ordóñez, Patricia 1919, 1936
Ortega, Courtney 632, 826
Ortiz, Ariel 1531
Ostrander, Tina J..... 456, 1937
Oswal, Hitender K. 1774
Oswal, Sushil K..... 1774
Otero, Salome 972
Ottenbreit-Leftwich, Anne T. 1710
Ouni, Ali 60
Ovabor, Kelvin 1887
Overcast, Levi 1889
Owen, Charles B..... 812
Owen, Rosalind 1688, 1690, 1776
Padiyath, Aadarsh 1000
Palmer, Jen 1335
Panciera, Katherine 1938
Pang, Ashley 1354, 1361, 1624
Pannuto, Pat 923
Paradise, Orr 352
Paredes-Robles, Carlos 951
Parham-Mocello, Jennifer 1007
Park, Hyejin 1014
Park, Hyesung 283
Park, Sunjae 1778
Parker, Miranda C. 1021
Parkinson, Jack 1028
Patel, Pankati 1035
Paul, David 1161
Paul, Jody 1918
Pavlovich, Tina 611
Payton, Jamie 1456
Paz, Jed 1746
Pearce, Hammond 1314
Pearson, Kim 1782
Pearson, Tamara 151, 1042, 1539
Pektaş, Emrah 1548
Peña, Joslenne 1919
Perdriau, Christopher 1049, 1652
Perez, Victor 1857
Pérez Quiñones, Manuel A..276, 806, 1900, 1915, 1919
Peric, Aleksandra 687
Perkins, Mckay 1126
Perrone Smith, Corrina 1904
Petersen, Andrew 1237, 1477, 1578, 1730
Petersen, Marianne Graves 902
Petrecca, Joe 1882
Petrulis, Robert 1554
Pexa, Abbigail 1656
Pfeffer, Jürgen 484
Pham, Kathy 1935
Pham, Tony 1588
Phatak, Jaai Uday 1929
Phelps, Victoria 1924
Pias, Marcelo 1897
Piech, Chris 583, 736, 785, 1395, 1442, 1844
Pierson, Timothy J..... 611
Pinkard, Nichole 1098
Pitt, Leonard 562
Pitts, Griffin 1887
Plane, Jandelyn 694

Pohlmann, Tyler 1656
Poitras, Eric 1808
Polak, Monika 137
Politz, Joe Gibbs 109
Pontelli, Enrico 1840, 1877
Popov, Aleksandr 1942
Porter, Leo 512, 1168
Posadas, Brianna 1919
Poskitt, Christopher M. 1307
Posner, Michael A. .. 1702, 1920, 1937
Postner, Lori 144, 1196, 1780
Poulsen, Seth 1056, 1063, 1722
Powley, Wendy 1920
Prasad, Prajish 1070
Prasad, Priya V. 1428
Prasad, Sushil 1906
Prather, James .296, 1063, 1521, 1916
Price, Thomas W. 526, 1402, 1792, 1818
Price, Thomas 1904
Pulimood, S. Monisha 1782
Pulling, William 1913
Qi, Jianzhong 1175
Qu, Danni 1883
Quam, McKenna 729
Quiroz, Nicolas 498
Quiroz, Patricio 1077
Rachmatullah, Arif 1804
Radziszowski, Stanisław 137
Rafferty, Anna N. 979, 1084
Rahal, Imad 1656
Rahman, Akond 1784
Raj, Rajendra K. 1867
Ramagoni, Sujeeth Goud 1786
Ramos, Kyla 1882
Ranade, Gireeja 1752
Randolph, Tim 1091
Rangel, Alessandra 1582, 1608, 1788, 1800
Rankin, Yolanda A. . 1098, 1539, 1638
Rantanen, Esa 1556
Rattigan, Matthew 1342
Ravi, Yashas 1942
Ravishankar, Veena 1790
Rayavaram, Pranathi 1105, 1740
Reckinger, Shanon 1112
Reddy, Mallika 1700
Redfield, Kamen 1119
Reed, David 1541
Reeves, Brent N. 296, 1063
Regatti, Nandika 1272
Reich, Justin 972
Reichert, Heidi 1792
Reid, Karen 1916
Reilly, Cole 1342
Ren, He 1021
Ren, Zixiao 1668
Renzella, Jake 1314, 1692, 1714
Repenning, Nadia 1904
Rich, Kathryn M. 1126, 1768
Richards, Ethan 1794
Richards, Timothy 207
Ricks, Vance 1921
Riley, Derek 1938
Ritz, Anna 1884
Rivera, Elijah 1133, 1917
Rivera, Veronica A. 1140
Roberson, Christian ..186, 1900, 1915
Robinson, Christa 701
Robledo Yamamoto, Fujiko 470
Roca, Rachel 1941
Rogers, Michael P. 1147
Ronan, Darcy 1570, 1796
Rorrer, Audrey 276, 833, 1456
Ros, Kevin 1798
Rosales, Jennifer 1154
Rosato, Jennifer 1590, 1910
Rose, Rachel 1
Rosenberg, Joshua 1935
Ross, Anne Spencer 324
Roy, Nimisha 1932
Rudolph, Carsten 853
Rudy, Caitlin 694
Ruiz, Pati 1608, 1788, 1800
Rusch, Ana Mary 1802
Russell, Seán 46
Rutstein, Daisy 1804
Sabin, Mihaela 1896
Sachithanand, Krithi 1941
Saffar Perez, Mariam 1652
Sahin, Ferat 1556
Sahin, Mehmet 687
Saiani, Andrea 1728
Saito, Daisuke 1838
Saito-Stehberger, Dana 1806
Sakamoto, Kazunori 569
Sakr, Majd 116, 1550, 1826, 1842
Sakzad, Amin 853, 1161
Salac, Jean 1712
Salas, Erika 1740
Salem, Tawfiq 1859, 1861
Salguero, Adrian 1168
Saliba, Liam 1175
Salloum, Mariam 450, 1664
Sampangi, Raghav V. 1808
San Martín, Daniel 1750
Sanchez, Maria 576
Sandoval, Juan Pablo 951
Sandu, Ana Smaranda 1865
Sane, Aamod 1070
Sanusi, Ismaila Temitayo 792, 1871
Sarsa, Sami 1063
Savelka, Jaromir 1550, 1842
Savelka, JaromÍr 116
Sax, Linda J. 1182
Scalisi, Corrie 1925
Scharks, Tim 1937
Schechter, Ariel 1890
Schiff, Daniel 1560
Schiller, Kathryn 1855
Schmith, Alex 986
Schonfeld, Paul 1590
Schrock, Tyler 456
Schroeder, Rebecca 1938
Schulte, Carsten 443
Schwartz, Jordan 1810, 1834
Schwartz, Matt 1928
Schwarzhaupt, Robert 1812
Scott, Allison 1543
Scurry, Briana 1873
Selby, Justin 1882
Selva, Manuel 1189
Servin, Christian 1196, 1943
Sexton, Stacey 1529
Seyam, Mohammed 338
Shaffer, Cliff 338, 867, 1904
Shah, Anshul 1202, 1209
Shah, Meghan 881
Shah, Rahul 860
Shah, Sheikh Ahmad 1929
Shahriar, Hossain 1784
Sharma, Priya 25
Sharmin, Sadia 708
Shaw, Mia S. 1539
She, Yunyi 1216
Sheard, Judithe 853, 1161
Sheard, Judy 1223
Shekhar, Prateek 1546
Sheldon, Josh 972
Shen, Yiyin 1230

Sherwood, Heather 1814, 1816
Shi, Yang 526, 1640, 1818
Shilton, Katie 722
Shinohara, Kristen ... 324, 1556, 1892
Shioji, Elisa 1175
Shook, Doug 1905
Shoshitaishvili, Yan 930, 937
Shumar, Wesley 144, 1780
Sibia, Naaz 1237, 1477, 1578
Siddappa, Sara 1537
Sidhu, Sukham 1119
Siever, Bill 1921
Silberglitt, Matt 1688
Silva, Leonardo 1244
Silva, Mariana 303, 317, 359, 429, 1909
Silva de Oliveira, Gabriel 1251, 1646
Silvia, Devin 1941
Simion, Bogdan 1477
Simon..... 799, 1161, 1223
Singh, Anjali 1258
Singh, Pawanjeet 116
Singh Walia, Gursimran 1939
Singla, Adish 387
Skerritt, Matthew P. 1161
Skripchuk, James 1818
Skuratowicz, Eva 540
Smith, Derrick 1300
Smith, Eric 1477
Smith, Gillian 1265
Smith, Jacqueline 1525
Smith, Julie M..... 972, 1820, 1822
Smith, Justin 53
Smith, Katie N. 715
Smith IV, David H. 1824
Snider, Johan Mattias 1888
Snowden, Michael 1930
Soares, Gustavo 1258
Sobiesk, Edward 248
Solomon, Carly 1888
Sommer, Nathan 1706, 1914
Sommers, Emma 909
Song, Yang 255
Song, Yifan 1826
Song, Yukyeong 1272
Sonka, Kate 1892
Soosai Raj, Adalbert Gerald 512, 618, 1202, 1209, 1230, 1381
Souza Teixeira, Erico 1892
Spang, Marissa 1126
Sparmann, Sören 443
Specht, Marcus 262
Speer, Annika 450
Spertus, Ellen 1905
Spezialetti, Madalene 1828
Spindt, Sonia 1794
Sprenger, Markus 1279
Sproull, Todd S. 1905
Sridhar, Pragnya 1550
Srinivasa Ragavan, Sruti 67
Starke, Elena 484
Stefik, Andreas 1286, 1915
Steinhorst, Phil 1293
Stephens, Heather 1902
Stephens-Martinez, Kristin 1634
Stewart, Michael C. 1922
Sthapit, Sandeep 1792, 1830
Stoner, Joshua 1879
Stormes, Kaitlyn N..... 715, 1182
Stoycheva, Milena 1728
Strange, Laney 1865, 1911
Streator, Joycelyn 1935
Stricklan, Chris 986
Strickland, Carla 1533
Su, Norman Makoto 1140
Su, Tong 708
Su, Zhendong 1321, 1328
Sudhakaran, Sneha 986
Sultana, Simon 1939
Summet, Valerie 1922
Susanto, Nicholas 1578
Sussman, Alan 1906
Syamkumar, Meenakshi 1230
Szabo, Claudia 1871
Szafir, Daniel 1676
Tabarsi, Benyamin T. 1792
Tabassum, Moumita 1300
Tack, Guido 853
Tadimalla, Sri Yash 1529
Tagare, Deepti 1896
Tan, Ivan 1307
Tan, Jialiang 1832
Tan, Yinyue 1237
Tan, Youxin 1668
Tandon, Seth 1774
Tang, Cara 1196, 1913
Tang, Yuerou 1810, 1834
Tanimura, Ellie 200
Tareque, Md. Hasan 1836
Tartaro, Andrea 1541
Tate, Carol 1804
Tate, Rhonda 1698
Taylor, Andrew 1314
Taylor, Cynthia 1908
Teehan, Ryan 53
Temitayo Sanusi, Ismaila 1740
Tey, Joseph 200
Thacker, Ian 1938
Thai, Justin 1598
Thayer, Kyle 1929
Thomas, Debra “Kelly” 1566
Thomas, Jakita O..... 1098
Thomas, Madison 1830
Thomas-Cappello, Elizabeth 95
Thompson, Clarissa 1456
Thorgeirsson, Sverrir 1321, 1328, 1929
Thornton, Patrick 750, 1928
Tian, Ruochen 1838
Tian, Xiaoyi 1272
Tise, Joseph C. 819
Tiwari, Ashish 1258
Toker, Sacip 1552
Tong, Thanh 1209
Torbey, Ryan 1812
Torres Castillo, Ruth 1840, 1877
Townsend, Gloria 1920
Tralie, Christopher John 1943
Tran, Keith 463, 1818
Tran, Minh 1335
Traversaro, Daniele 1622
Treude, Christoph 366
Trim, Michelle 1342
Trinh, Conroy 1668
Tseng, Ying-Jui 1842
Tsuji, Ayumi 1838
Tuck, Anthony 1342
Tucker, Cindy S..... 1196
Tuohy, Brenda 632, 826
Twarek, Bryan 1822
Twomey, Robert 1926
Tychonievich, Luther A... 1873, 1895
Tymann, Paul 1535
Uchidiuno, Judith 1560, 1562, 1718, 1720
Uma, R.N. 1865
Unfried, Alana 1702, 1937
Urry, Joshua 1347
Vaccaro, Kristen 109
Vadarevu, Anagha 477, 1670

Vahedian Movahed, Saniya 1740
Vahid, Frank 1354, 1361, 1368, 1624, 1869
Vahrenhold, Jan 1293
Vaidyanathan, Ramachandran . 1906
Valdez, Tomas 1840, 1877
Valentine, Andrew 366
VanDeGrift, Tammy 1374
Vandenberg, Jessica 1660, 1680, 1718
Vanderberg, Maggie 540
Vásquez, Andrea 1750
Vassar, Alexandra 1314, 1714
Veilleux, Nanette 1885, 1922
Veliz, Oscar 1919
Vellamchety, Krishna 1105
Verdicchio, Michael 179
Villegas Molina, Ismael ... 1168, 1381
Vo, Lily 1498
Voggenreiter, Angelina 484
Voida, Amy 269, 470
Von Vacano, Claudia 576
Vrbik, Paul 757
Waisome, Jeremy A. Magruder . 1857
Walker, Justice Toshiba..... 1527
Walther, Kendra 1911
Wan, Zipiao 1550
Wang, Bingcheng 1237
Wang, Chun 1021
Wang, Jingwen 1879
Wang, Kevin 456, 1388
Wang, Matt X. 1525
Wang, Nina 533
Wang, Sierra 1395, 1844
Wang, Wei 1766
Wang, Wengran 1402
Wang, Xu 1686
Wang, Yuanxin 1826
Wang, Zhuoying 1409, 1910
Ward, Paul A.S. 1596
Warner, Jayce 151
Warschauer, Mark 1648, 1806
Washburn, Max 826
Washington, Talitha 1760
Washizaki, Hironori 569, 1838
Watkinson, Neftali 450, 1664
Weber, Jason Lee..... 1846
Weems, Charles 1906
Weese, Josh 1907, 1910, 1928
Wehrwein, Scott 944
Wei, Wei 1848
Weidmann, Karl-Heinz 1328
Weidmann, Theo B. 1321, 1328, 1929
Weill-Tessier, Pierre 165
Weinberger, William 1851
Weinman, Nathaniel 1853
Weintrop, David 227, 694, 1335, 1527
Weiqi, Victor Huang 1414
Weiss, Mark 74, 1932
Weng, Shiny 860
Wermelinger, Michel 1916, 1930
Werner, Stephanie M. 1421
West, Leah 993
West, Matthew 429, 1056, 1909
Westerlund, Jill 1875
Whitaker, Douglas 1702, 1937
White, Jonathan 847
White, Pauline 1855
White, Shana 1533, 1543
Wick, Jonathan 1774
Wiese, Eliane 979
Wilde, Caitlyn 1889
Wilde, Jina 1428
Wilkinson, Michael 1890
Williams, Chad A. 1706, 1914
Williams, Grace 1722
Williams, Heidi 1796
Williams, Joseph Jay 1237, 1477
Wilson, Joseph P. 1126, 1812
Wilson, Linda F. 1900, 1915
Wolfman, Steven 1736
Wolkowicz, Kelilah 1740
Won, Myounggyu 1435
Wong-Ma, Jennifer 1742, 1846
Wood, Zoë 646
Woodrow, Juliette 736, 785, 1442
Wortel-London, Stephanie B. 221
Wright, Ian 1890
Wright, Kerene 352
Wright, Melissa 1154
Wright, Rebecca N. 1154
Wright, Simone 604
Wu, Amy 1857
Wu, Andrew C. 200
Wu, Fan 547, 1784
Wu, Longfei 1893
Wu, Wensheng 1449
Wu, Ying Choon 1926
Wu, Zihan 1662
Wünsche, Burkhard C. 519
Wurst, Karl R. 144
Wyatt, Lauren Gabrielle 1456
Xiang, Lili 1890
Xiang, Qiao 1897
Xiao, Ruiwei 1842
Xiao, Yanbing 88
Xing, Guangming 1859, 1861
Xu, Iris 1736
Xu, Jessica Jia-Ni 757
Xu, Shaoxuan 1584
Xu, Shengsong 1668
Xu, Weiyin 1891
Yadav, Aman . 1535, 1543, 1914, 1934
Yamaguchi, Ryoko 1463
Yamamoto, Ryuta 1279
Yan, Lisa 860, 1810, 1834
Yan, Wei 1863
Yang, Austin 1668
Yang, Hui 1804
Yang, Ke 1939
Yang, Ruhan 1930
Yarnall, Louise 1894
Ye, Runlong 1477
Yim, Jacob 1810, 1834
Yoon, Ilmi 373
Yu, Jerry 1209
Yuan, David 736
Yuen, Timothy T. 1428
Zakrajsek, Todd 3
Zarch, Rebecca 151
Zarkoob, Hedayat 1470
Zavaleta Bernuy, Angela 1237, 1477, 1578
Zawatski, Michael J. 1428
Zegura, Ellen 1560, 1562
Zeitz, Jessica 1790
Zeller, Amanda 888
Zenke, Carter 750, 1902, 1903, 1916, 1927, 1928, 1940
Zhai, ChengXiang 1798
Zhang, Amy 123
Zhang, Helen 1929
Zhang, Jiayi 214
Zhang, Liang 1484
Zhang, Lisa 1521
Zhang, Riteng 1883
Zhang, Runqiu 255
Zhang, Xiaoling 172
Zhang, Yue 1784

Zhou, Fangtong 255
Zhou, Iris 88
Zhu, Tingting 1730
Zhukovets, Yuliia 1940
Zilles, Craig 303, 317, 429, 1824, 1909
Zimmer, Michael 1921
Ziv, Hadar 1846
Zong, Yue 1484
Zuckerman, Austin L. 1491, 1498