Innovative Courses that Broaden Awareness of CS Careers and Prepare Students for Technical Interviews*

Jean Griffin[1], Legand Burge[2], Sally Goldman[1], Diego Aguierre[3], Juan Alonso Cruz[1], April Alvarez[1], Albert Cervantes[1], Shameeka Emanuel[1], Ann Gates[3], Daniel Gillick[1], Christopher Hogan[1], Jeremy Hurwitz[1], Janaki Lahorani[1], Mary Jo Madda[1], Olumid Malomo[2], Jenn Marroquin[1], Nisha Masharani[1], Alycia Onowho[2], Angela Pablo[1], Jason Randolph[1]
[1]Google, Mountain View, CA
{jeangriffin, sgoldman}@google.com
[2]Howard University, Washington, D.C.
{lburge, alycia.onowho}@howard.edu
[3]University of Texas at El Paso, El Paso, TX
{agates, daguirre6}@utep.edu

Abstract

While undergraduate Computer Science (CS) degree programs typically prepare students for well-established roles (e.g. software developer, professor, and designer), several emergent CS career roles have gained prominence during the 21st century. CS majors (and students considering CS as a major) are often unaware of the wide range of careers available to job candidates with a CS background. This experience report describes seven innovative courses that broaden awareness of CS career roles and prepare students for technical interviews. Five courses

prepared students for these career roles: Full-Stack Developer, Product Manager, ML or NLU Scientist, Technical Entrepreneur, and User Experience Designer/Developer/Researcher. The other two courses had traditional content but explicitly prepared students for technical interviews. These courses were co-developed by industry professionals and CS professors, and co-taught during a semester-long academic program. This paper highlights the replicable aspects of the program: the courses, teaching practices, and evaluation instruments (a teaching practices inventory and a data structures inventory).

1 Introduction

It is well known that Computer Science (CS) majors often have gaps in career readiness when interviewing for jobs in the tech industry and during their first months on the job [4, 13, 14, 19, 20, 24, 25]. Another critical but lesser-known gap is in career awareness. CS degree programs prepare students for established roles (e.g. software developer, professor, designer) but typically aren't as well equipped to prepare them for newer roles such as Full-Stack Developer, Product Manager, Machine Learning (ML) or Natural Language Understanding (NLU) Scientist, Technical Entrepreneur, or User Experience (UX) Professional.

This paper addresses gaps in career awareness by describing five courses taught in 2020 that introduced undergraduate CS majors to these careers. It addresses gaps in career readiness by describing how two additional courses explicitly prepared students for technical interviews. It also describes the program's teaching philosophy, which was designed to foster learning, soft skills, and social capital building. Although the courses were taught within a program sponsored by an industry-academic partnership, this paper focuses on the replicable aspects of the program (courses, teaching practices, inventories) rather than the partnership itself. This information is useful for departments and other programs that seek to address gaps in career awareness or career readiness through courses, events (e.g. career panels), or interview prep clubs.

This paper is organized as follows. The background section describes the program's setting, participants, and teaching philosophy. The next two sections discuss the courses that broadened awareness of CS career roles, and the courses that provided interview preparation. The paper concludes with a discussion.

2 Background

2.1 Setting and Participants

The courses described in this paper were taught during the 2020 Spring semester within an academic program hosted by an industry-academic partnership. The

partnership involved Google, a host university that awarded academic credits (Howard University), and nine additional colleges/universities. The program was designed to be a residential, immersive experience for students and visiting faculty members to spend a semester on the Google campus taking (or co-teaching) CS courses. In keeping with a design-based research methodology [7, 27], the 2020 program built on lessons learned from other recent industry-academic partnerships for CS, e.g. [3, 11, 15, 21] and prior iterations of the program [2]. The courses were chosen by balancing the availability of Google employees to co-develop and co-teach courses pertaining to their profession (e.g. Full Stack Developer), and the need to offer some core courses so students could satisfy degree requirements (e.g. Database Systems).

The 2020 Spring participants included 40 undergraduate CS majors (40% female), 12 instructors, and more than 100 Google volunteers who served as mock interviewers, teaching assistants, mentors, guest speakers, and panelists. Students took all of their Spring courses within the context of this program, choosing to take at least four of the seven courses offered.

Details about the industry-academic partnership, residential life, and community-building were discussed in [2]. While these aspects of the program would be difficult to replicate under normal circumstances, the 2020 program was interrupted by COVID-19 and switched abruptly to remote learning mid-semester. Subsequent iterations of the program have been fully remote. The goal of this paper is to focus on the the widely replicable aspects of the 2020 program: the courses, teaching practices, and inventory instruments.

2.2 Teaching Philosophy

The teaching philosophy that served as a guiding framework for the program was informed by the work of Carl Wieman, the Nobel prize winner and Stanford professor who founded the Science Education Initiative to research and recommend effective evidence-based teaching practices. The Teaching Practices Inventory (TPI) is an outcome of this work [29, 28, 16]. The inventory's eight categories of effective teaching are: 1) Course information provided to students via hard copy or course web page; 2) Supporting materials provided to students; 3) In-class features and activities, especially active learning; 4) Assignments; 5) Feedback and testing; including grading policies; 6) Other (e.g. surveys and other evaluation instruments); 7) Training and guidance of Teaching Assistants; 8) Collaboration or sharing in teaching.

The 2020 program addressed the TPI categories as follows. Categories 1+2) A learning management system served as a communication vehicle and hosted all the resources necessary for students to succeed; 3) Active learning was used extensively (discussed below); 4+5) Students had frequent assignments with timely feedback starting no later than the third week; 6) Evaluation instru-

ments included mid-semester and end-semester course feedback surveys, an attitudes survey, and the Basic Data Structures Inventory [23]; 7) Course team members attended an orientation focused on active learning; 8) Each course team met weekly; an additional weekly meeting included all the course teams; classroom observations were conducted throughout the semester, which generated feedback for the instructors.

Active learning warrants special mention. With active learning, students participate in the class and become a part of the learning process. Students are engaged as opposed to being passive, e.g. while listening to lectures. When managed effectively, active learning promotes learning [28]. The instructors employed active learning in a variety of ways, usually constraining lectures to 5-20 minutes. All the courses used small group learning in various forms. In most courses, student teams presented reports or demos during class time. All the instructors used whiteboarding extensively, facilitated by floor-to-ceiling whiteboards. One technique involved "everybody up!" where all students got up to work on whiteboards in small groups while the instructors observed and moved from group to group. The Technical Entrepreneurship class used a flipped classroom approach. The Machine Learning class used slide decks that had periodic slides with prompts for small group discussion. All had guest speakers where students had the opportunity (or requirement) to ask questions of the speaker.

Since active learning engages students with communication and collaboration, it supports the development of soft skills. It also fosters social capital building. Social capital is the relationships and networks one has within a community, such as the tech community. Social capital building expands one's network, fosters a sense of belonging, and shapes one's identity as a professional [6]. The importance of social capital for success in the tech industry, especially for students from historically marginalized groups, is often overlooked [9].

3 Courses that Broadened Awareness of CS Career Roles

Each section below focuses on a career role and the course that introduced it.

3.1 User Experience Designer/Developer/Researcher: HCI Course

Many CS departments now offer Human Computer Interaction (HCI) courses, but they are often electives. As a result, many CS majors are unaware of the variety of User Experience (UX) careers within this field including Designer, Developer/Engineer, and Researcher. UX professionals conduct research to understand people, their needs, and their contexts. They apply that knowledge to design, test, communicate, and deliver easy-to-use technologies. They work

closely with Product Managers and Engineering teams to identify problems and opportunities, and iteratively develop solutions.

The UX instructors provided hands-on experience with the processes practiced at Google to design and build products. Students learned about developing solutions through the lens of accessibility, inclusive design, and equity. This focus on underrepresented and vulnerable users empowered students to design with communities, to consider broader systems and contexts as they defined problems, and to create innovative solutions to serve broader populations and deliver equitable experiences.

3.2 Full-Stack Developer: Software Design Studio Course

CS degree programs typically require a Software Engineering (SE) course, but many SE courses and textbooks do not cover full-stack software development. The past decade has seen rapid advances in web technologies, resulting in a high demand for full-stack developers. Privileged students with strong networks can learn these skills outside of the classroom, but many students do not have such networks. The Software Design Studio course provided this training.

In this course, teams of 4-5 students engaged in the software development lifecycle by designing and building a fully functional web app. Each team produced designs, milestones, code, and stretch goals. They were required to use the same JavaScript front-end library (React) and backend service (Firebase). Otherwise, they had considerable flexibility to create a web app based on their own interests. Guest speakers lectured on topics including security and privacy, site reliability engineering, and real-world case studies. At the end of the semester, each team gave a live demo of their app, and discussed how it was built and how the team functioned.

3.3 ML or NLU Scientist/Engineer: ML Course

The field of machine learning (ML) leapt to prominence during the last decade. Now ML research scientists and engineers are in high demand. This ML course provided a gentle introduction to the field; the only prerequisites were an introductory Data Structures course and experience with Python. It touched on many applications of ML from everyday life such as recommendation systems and sentiment analysis, along with data visualization tools. Students learned about classification, regression, and end-to-end ML pipelines. Weekly assignments often involved adapting supplied code that used pandas and TensorFlow libraries within Colab notebooks. At the start of the semester, most students already had a sense of what ML is and how it is used. In contrast, most were unfamiliar with the related field, Natural Language Understanding (NLU), even though they are familiar with many apps that use NLU – to

translate languages, auto-complete sentences, and answer questions within a web browser. The social implications of ML were discussed throughout the course in small-group and whole-class discussions. A member of Google's ML Ethics team led a full class period dedicated to issues of ethics, bias, and fairness. The final team project involved solving a real-world problem in a Kaggle competition.

3.4 Product Manager: PM Course

Although Product Managers (PMs) often have leadership roles in tech companies, undergraduate CS majors are not necessarily exposed to Product Management as a discipline. Many PM classes are offered by business programs, such as *Product Management 101 and 102* at Harvard Business School. PMs help engineering and design teams understand what their target users need, drive effective collaboration within the team to design and build a solution, evaluate the success of that solution, and iterate and expand upon it. The PM course was designed to help students develop the skills needed to succeed in an entry level PM role. In a group project, they assumed the role of the PM through the product development lifecycle for a product of their own invention. Students developed soft skills through negotiation, collaboration, communication, and learning to exert influence without authority.

3.5 Technical Entrepreneur: Technical Entrepreneurship Course

Successful entrepreneurs need to be innovative and have good business, communication, cognitive design, and leadership skills. In the Technical Entrepreneurship course, teams within a flipped classroom were immersed in the startup culture. The goal for each team was to develop a business model and produce a product solution. Students participated in customer discovery and engaged in an iterative, fast-paced engineering design-build-test loop. They interacted with Google engineers and local entrepreneurs who served as subject matter experts, mentors, guest lecturers, and co-instructors. At the end of the semester, each team pitched their product to the class and guests from the community.

4 Courses That Provided Interview Preparation

This section discusses the importance of explicit technical interview preparation and how two courses provided this preparation.

4.1 The Importance of Technical Interview Preparation

Technical interviews for CS internships and full-time jobs often focus on data structures and algorithms. Undergraduate CS degree programs typically require a Data Structures (DS) course, which students typically take during their first or second year, e.g. [1]. Despite this preparation, many candidates struggle to pass technical interviews. Contributing factors include content knowledge, practice with problem solving, knowledge of interview norms, and anxiety. Although there are several good, well-known resources that help candidates prepare for technical interviews through self-study [22], fostering a community of practice with peers, e.g. in a course [18] or club, is desirable.

Regarding content knowledge, the coverage of topics in a DS course varies from school to school. Some topics that are frequently addressed during technical interviews are not covered in all DS courses, such as hash tables and runtime complexity [23]. Students often have insufficient practice with thinking about how and when to apply DS concepts to solve problems. Without a good foundation in problem solving, students often don't even know where to begin. Many DS courses only provide blocked practice (learn topic A and practice topic A; learn topic B and practice topic B; and so on). In contrast, distributing practice with a topic over time, where the topics are interleaved, is more effective [8, 26, 30]. Ample effective practice develops the mental flexibility to evaluate the pros and cons of various solutions and deal with the ambiguity that is typical in a technical interview.

Regarding familiarity with the interview process, it is helpful if the candidate is aware of interview norms. It is often acceptable to ask for clarifications; interviewers may pose problems that are intentionally tricky or unsolvable; and it is common for professionals to undergo multiple interviews, even with the same company, before landing an offer. It is important for candidates to be comfortable with solving problems on a whiteboard. Whiteboarding is typically used during technical interviews – where candidates write code on a whiteboard while the interviewer poses questions, observes the candidate's process, and provides help. Sometimes pair programming is used.

Researchers who studied anxiety in technical interviews found that mock interviews can instill confidence and reduce anxiety. They suggest that companies should understand how anxiety may negatively impact interview performance, and that CS departments should help students prepare for them [17]. Although anxiety and impostor syndrome affect many interview candidates, it can be compounded for people from historically marginalized groups, many of whom are unfamiliar with the norms of technical interviews [12, 9, 10].

4.2 Courses That Provided Technical Interview Preparation

Two courses explicitly prepared students for technical interviews. Both had an introductory Data Structures course as a prerequisite.

The Applied Data Structures course focused on practice with topics from a basic introductory Data Structures course [23] and other topics that students may encounter during a technical interview such as hash tables and runtime complexity. Students spent much of class time solving problems on whiteboards. At the start of the semester the Basic Data Structures Inventory (BDSI) was administered as a pre-test (in-person, on paper) to gauge prior knowledge [23]. Because their prior knowledge varied considerably, the course employed a mastery learning approach. Students mastered topics at their own pace, working in small groups with peers at the same level [5]. To provide authentic practice, Google volunteers conducted 210 mock interviews.

In the Database Systems course, beyond covering topics such as good data design and SQL programming, students learned how data structures and algorithms are used within database systems. Activities and assignments engaged students with problems encountered in technical interviews and the real world.

5 Discussion

At the end of the semester, 36 of 40 students (90%) completed a non-anonymous online survey. Key takeaways from the 5-point Likert scale questions are: 95% rated their overall experience with the program as positive; 97% felt they became better programmers; 91% felt likely to succeed in a high-tech job; and 94% reported that their soft skills improved. In response to the question "What did you find especially valuable about your academic experience that was different from your prior university experience?" some representative answers included: 1) "This program was nothing like what I've experienced through my years in school. Most of the courses I've taken here aren't offered at all at my university;" 2) "Now, I have work to show and full project experience;" 3) "I think the mock interviews were probably one of the best things for me because I've never experienced one before and now I am comfortable when I do have one."

A notable limitation to this work was triggered in March 2020 when the COVID-19 pandemic prompted a switch to online classes, which students attended from home or another remote location. The data structures post-test could not be given because the BDSI can only be administered in-person and on paper [23]. Comparing pre-test and post-test results would have provided valuable information about the program with respect to interview preparation.

This paper provides valuable insights about professional preparation for CS students. It describes innovative courses that introduce students to lesser known but exciting careers. It provides valuable insights for preparing students

for technical interviews. It describes effective teaching practices that promote learning, soft skills, and social capital building. It also describes two evaluation instruments – a teaching practices inventory and a data structures inventory. The information shared can be adapted by CS departments and other programs to address gaps in career awareness and career readiness through coursework, events (e.g. career panels), or interview prep clubs.

References

[1] ABET Computing Accreditation Comission. *Criteria for Accrediting Computing Programs, 2020 – 2021*. Baltimore MD, 2019.

[2] April Alvarez et al. "Google Tech Exchange: An Industry-Academic Partnership That Prepares Black and Latinx Undergraduates for High-Tech Careers". In: *Journal of Computing Sciences in Colleges* 35.10 (Apr. 2020), pp. 46–52.

[3] Matthew Barr and Jack Parkinson. "Developing a Work-based Software Engineering Degree in Collaboration with Industry". In: *Proceedings of the 1st UK & Ireland Computing Education Research Conference*. ACM, 2019, pp. 1–7. ISBN: 1595930361.

[4] Andrew Begel and Beth Simon. "Struggles of new college graduates in their first software development job". In: *Proceedings of the 39th ACM Technical Symposium on Computer Science Education (SIGCSE '08)*. ACM, 2008, pp. 226–230. ISBN: 9781595937995.

[5] Benjamin S Bloom. "Learning for Mastery". In: *UCLA Evaluation Comment* 1.2 (May 1968), pp. 1–12.

[6] P. Bourdieu. "The forms of capital". In: *Handbook of theory and research for the sociology of education*. Ed. by J. G. Richardson. New York NY: Greenwood Press, 1986, pp. 241–258.

[7] Ann L Brown. "Design Experiments: Theoretical and Methodological Challenges in Creating Complex Interventions in Classroom Settings". In: *Journal of the Learning Sciences* 2.2 (1992), pp. 141–178.

[8] Peter C. Brown, Henry L. Roediger III, and Mark A. McDaniel. *Make it Stick: The Science of Successful Learning*. Cambridge MA; London England: The Belknap Press of Harvard University Press, 2014.

[9] Q. Bui and C. Cain Miller. "Why Tech Degrees Are Not Putting More Blacks and Hispanics Into Tech Jobs". In: *The New York Times* (Feb. 2016).

[10] Legand Burge, Katherine Picho-Kiroga, and Portia Kibble Smith. *The Interview Access Gap for Black Engineers*. Tech. rep. Seattle, Washington: Karat, 2021. URL: https://karat.com/wp-content/uploads/2021/09/The-Interview-Access-Gap-for-Black-Engineers-v2.pdf.

[11] Gail Carmichael et al. "Curriculum-Aligned Work-Integrated Learning". In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM Press, 2018, pp. 586–591. ISBN: 9781450351034.

[12] K. Cokley et al. "Impostor feelings as a moderator and mediator of the relationship between perceived discrimination and mental health among racial/ethnic minority college students". In: *Journal of Counseling Psychology* 64.2 (2017), pp. 141–154.

[13] Denae Ford et al. "The tech-talk balance: what technical interviewers expect from technical candidates". In: *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE Press, 2017, pp. 43–48. ISBN: 9781538640395.

[14] Vahid Garousi et al. "Closing the Gap Between Software Engineering Education and Industrial Needs". In: *IEEE Software* 37.2 (2019), pp. 68–77. ISSN: 19374194.

[15] Kinnis Gosha et al. "Strategic partnerships to enhance data structures and algorithms instruction at HBCUs". In: *Proceedings of the 2019 ACM Southeast Conference (ACMSE 2019)*. ACM, 2019, pp. 194–197. ISBN: 9781450362511.

[16] Mark Guzdial. "How I Evaluate a College Computer Science Teaching Record". In: *Communications of the ACM* (Feb. 2021).

[17] Phillip Hall and Kinnis Gosha. "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 (SIGMIS-CPR '18)*. ACM, 2018, pp. 64–69. ISBN: 9781450357685.

[18] Amanpreet Kapoor and Christina Gardner-McCune. "Introducing a Technical Interview Preparation Activity in a Data Structures and Algorithms Course". In: *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE (2021)* (2021), pp. 633–634.

[19] Wendy A. Lawrence-Fowler, Laura M. Grabowski, and Christine F. Reilly. "Bridging the divide: Strategies for college to career readiness in computer science". In: *Proceedings - Frontiers in Education Conference, FIE* 2014 (2015), pp. 1–8. ISSN: 15394565.

[20] Paul Luo Li, Amy J. Ko, and Jiamin Zhu. "What makes a great software engineer?" In: *Proceedings - International Conference on Software Engineering* 1 (2015), pp. 700–710. ISSN: 02705257.

[21] Joseph Maguire, Steve Draper, and Quintin Cutts. "What Do We Do When We Teach Software Engineering?" In: *Proceedings of the 1st UK & Ireland Computing Education Research Conference (UKICER '19)*. ACM, 2019, pp. 1–7. ISBN: 9781450372572.

[22] G. L. McDowell. *Cracking the coding interview: 189 programming questions and solutions*. CareerCup, 2019.

[23] Leo Porter et al. "BDSI: A Validated Concept Inventory for Basic Data Structures". In: *Proceedings of the 2019 ACM Conference on International Computing Education Research*. ACM, 2019, pp. 111–119. ISBN: 9781450361859.

[24] Alex Radermacher and Gursimran Walia. "Gaps between industry expectations and the abilities of graduates". In: *Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE 2013)*. ACM, 2013, pp. 525–530. ISBN: 9781450320306.

[25] Chris B. Simmons and Lakisha L. Simmons. "Gaps in the Computer Science Curriculum: An Exploratory Study of Industry Professionals". In: *The Journal of Computing Sciences in Colleges* 25.5 (2010), pp. 60–65.

[26] Nicholas C. Soderstrom and Robert A. Bjork. "Learning versus performance: An integrative review". In: *Perspectives on Psychological Science* 10.2 (2015), pp. 176–199.

[27] Feng Wang and Michael J. Hannafin. "Design-based research and technology-enhanced learning environments". In: *Educational Technology Research and Development* 53.4 (2005), pp. 5–23. ISSN: 1042-1629.

[28] Carl Wieman. "A Better Way to Evaluate Undergraduate Teaching". In: *Change: The Magazine of Higher Learning* 47.1 (Jan. 2015), pp. 6–15. ISSN: 0009-1383.

[29] Carl Wieman and Sarah Gilbert. "The Teaching Practices Inventory: A New Tool for Characterizing College and University Teaching in Mathematics and Science". In: *CBE Life Sciences Education* 13.3 (2014), pp. 552–569. ISSN: 19317913.

[30] Iman YeckehZaare, Paul Resnick, and Barbara Ericson. "A Spaced, Interleaved Retrieval Practice Tool that is Motivating and Effective". In: *Proceedings of the 2019 ACM Conference on International Computing Education Research (ICER '19)*. ACM, 2019, pp. 71–79.