Code, Connect, Create: The 3C Professional Development Model to Support Computational Thinking Infusion

Robin Jocius[1], Deepti Joshi[2], Yihuan Dong[3], Richard Robinson[2], Veronica Cateté[3], Tiffany Barnes[3], Jennifer Albert[1], Ashley Andrews[1], Nicholas Lytle[3]

[1]Zucker Family School of Education
The Citadel
Charleston, SC, USA
(rjocius, jalbert, ashley.andrews)

[2]School of Science and Mathematics
The Citadel
Charleston, SC, USA
(djoshi, rjmr)

[3]Department of Computer Science
North Carolina State University
Raleigh, NC, USA
(ydong2, vmcatete, tmbarnes, naltyle)

ABSTRACT

Despite the increasing attention to infusing CT into middle and high school content area classrooms, there is a lack of information about the most effective practices and models to support teachers in their efforts to integrate disciplinary content and CT principles. To address this need, this paper proposes the Code, Connect and Create (3C) professional development (PD) model, which was designed to support middle and high school content area teachers in infusing computational thinking into their classrooms. To evaluate the model, we analyzed quantitative and qualitative data collected from Infusing Computing PD workshops designed for in-service science, math, English language arts, and social studies teachers located in two Southeastern states. Drawing on findings from our analysis of teacher-created learning segments, surveys, and interviews, we argue that the 3C professional development model supported shifts in teacher understandings of the role of computational thinking in content area classrooms, as well as their self-efficacy and beliefs regarding CT integration into disciplinary content. We conclude by offering implications for the use of this model to increase teacher and student access to computational thinking practices in middle and high school classrooms.

CCS CONCEPTS

• Social and professional topics→ Computational thinking; K-12 education

KEYWORDS

Computational thinking; Professional development; K-12 computing; Teacher education

ACM Reference format:

Robin Jocius, Deepti Joshi, Yihuan Dong, Richard Robinson, Veronica Cateté, Tiffany Barnes, Jennifer Albert, Ashley Andrews, Nicholas Lytle. 2020. Code, Connect, Create: The 3C Professional Development Model to Support Computational Thinking Infusion. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11-14, Portland, OR, USA.* ACM, New York, NY, USA. 7 pages. https://doi.org/10.1145/3328778.3366797

SIGCSE '20, March 11-14, 2020, Portland, OR, USA.

ACM ISNB 978-1-4503-6793-6/20/03...$15.00
https://doi.org/10.1145/3328778.3366797

1 INTRODUCTION

As state and national standards for content area learning continue to evolve, computational thinking (CT), which involves "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" [27, p. 33], is rapidly becoming an integral piece of P-12 curricula (National Research Council, [19]). CT is not simply the integration of technology (e.g., use of computers or iPads) into P-12 education, but rather a set of practices and habits of mind that students can learn with or without the introduction of technology [20, 23]. Researchers [31] have even suggested that framing computational thinking as an essential component of disciplinary learning can support students' abilities to interact meaningfully with both computing and disciplinary content.

However, despite the increasing attention to integrating CT within K-12 content area instruction, there is little information available about the best ways to support teachers in developing and implementing learning experiences that explicitly target both disciplinary and content learning. In order to address this gap, this paper proposes the Code, Connect and Create (3C) professional development (PD) model, which is centered around three primary components—Code (Bootcamp), Connect (connecting disciplinary content and pedagogy to computational thinking), and Create (the development of CT-infused learning segments) (see section 3). The 3C model is developed as an essential component of a three-year research project, Infusing Computing, which aims to document how middle and high school teachers create and implement interdisciplinary, CT-infused lessons.

The 3C design and development process drew on the knowledge and experience of the research and facilitation team, which included computer scientists, education faculty who have worked as classroom teachers, in-service computer science teachers, and in-service content area teachers. Because teacher learning has often been a fragmented system [26], utilizing a situative perspective allowed us to examine not only what

teachers learn, but why they learn "to explore the connections among professional development activities and processes on the one hand, and individual teachers' knowledge and instructional practices on the other" [5, p. 7].

The 3C model has been implemented across two years of summer workshops that supported 116 middle and high school content area teachers in 2018 and another 150 teachers in 2019 in infusing computational thinking principles into their classrooms. This paper draws on qualitative and quantitative analyses of data from the Summer 2018 PD sessions and the 2018-2019 academic year implementation. Of the 116 teachers who attended summer workshops in 2018, 58 teachers that attended the North Carolina PD, 41% identified as science teachers, 36% identified as math teachers, and 22% identified as humanities teachers. Of the 58 teachers attending South Carolina PD, 33% identified as science teachers, 34% identified as math teachers, and 29% identified as humanities teachers. All 3C sessions were led by members of the research team with support from teacher leaders with computing infusion experience, including teachers who participated in the pilot study in which they infused CT into their classrooms. The results are discussed in section 4.

2 BACKGROUND

More than a decade ago, Wing's [27] seminal work described computational thinking as a "fundamental skill for everyone, not just for computer scientists" (p. 33). Since that time, there has been a growing interest in introducing CT in P-12 schools to support students' development of multi-faceted skills, such as critical, systematic thinking [15], scientific reasoning [21], mathematical practices [25], and expository writing [28]. The increasing availability of visual, block-based programming languages such as Scratch [17] and Snap! [14], as well as CT's potential to expand upon 21st century learning frameworks that emphasize problem-solving, innovation, and critical thinking [18], have been instrumental in creating an increased demand for CT integration into P-12 classrooms.

However, districts and schools continue to face significant challenges to CT integration, including a lack of clear policy guidance and logistical barriers to implementation. Currently, only 22 states have created computer science standards and only 15 states have adopted policies to ensure that all high school students have access to computer science (Code.org & CSTA, [8]). Researchers have discussed numerous solutions to overcome issues of access, including increasing the availability of after-school enrichment programs that emphasize programming [16], incorporating CT training into pre-service education courses [31], and providing training for teachers to teach AP Computer Science Principles and other computer-science-specific courses [1, 11]. While expanding CT practices into content classes has been proposed as a potentially valuable pathway for increasing access and broadening participation, mechanisms for introducing and supporting CT integration have been largely unexplored. As Grover and Pea [13] note, exploring the affordances for incorporating CT into disciplinary teaching, or "dovetailing the introduction of CT at K–12 with a transfer of problem-solving skills in other domains," has been an "under-investigated" area for both research and practice (p. 11).

Our work is grounded in the idea that computational thinking can be envisioned as an essential component of disciplinary learning. In order to accomplish the infusion of CT into disciplinary teaching and learning, teachers must integrate "activities that make visible the inherent overlap of computational thinking ideas and practices with subject area concepts," allowing students to interact meaningfully with both content and CT practices and skills [29, p. 567]. However, before any of these integration opportunities are possible, teachers must first understand what CT is, how it connects to their standards and curricula, and how it supports students' understandings of content.

Research has demonstrated that teacher professional development is critical to any successful change in educational policy or practice [22], including changes related to computational thinking integration into P-12 schools [3, 4]. Specifically, Barr and Stevenson [4] identified two major areas of need in relation to teacher PD in computational thinking: (1) a clear definition of what CT is and how it applies to students and content, and (2) explicit, ongoing training and support for P-12 teachers. To address these needs, the 3C model incorporates a reorganized definition of CT called PRADA, which is a practical way of introducing the big ideas of CT to non-computing teachers across disciplines to support them in infusing CT into their curricula [2]. PRADA is a mnemonic device that reorders and refines elements of Google's CT definition [12] to make the model more adaptable to different disciplines: Pattern Recognition: observing and identifying patterns and trends; Abstraction: identifying ideas that are important and relevant (giving them a name and hiding details); Decomposition: breaking down problems into meaningful smaller parts; and Algorithms: a composition of instructions for solving a problem and similar problems. These PRADA elements are used throughout each 3C session to provide support for teachers' developing understandings of CT and their design and development of CT-infused learning segments.

3 THE 3C MODEL

Within the 3C model, as teachers work towards the shared goal of creating learning segments that infuse computational thinking into their existing curricula, they expand on the models, vocabulary, forms of interaction, and participation structures within the computer science discipline. By the end of each week-long PD workshop, participants were expected to use their developing understandings of CT to design a lesson that involved the use of Snap! (2011), a programming language based on Scratch [17]. Teachers were tasked with creating the following components: (1) a Snap! prototype; (2) a detailed lesson plan; and (3) supplemental pedagogical materials, such as slides, links, or handouts. All elements of the 3C model were explicitly designed to scaffold teachers towards increasingly complex understandings of CT. Further, 3C aims to help teachers recognize integration opportunities in both discipline-specific and interdisciplinary ways. Figure 1 provides an overview of the 3C model, including the expected amount of time spent on each 3C session each day of

the PD workshop, facilitator characteristics, and sample activities for each component.

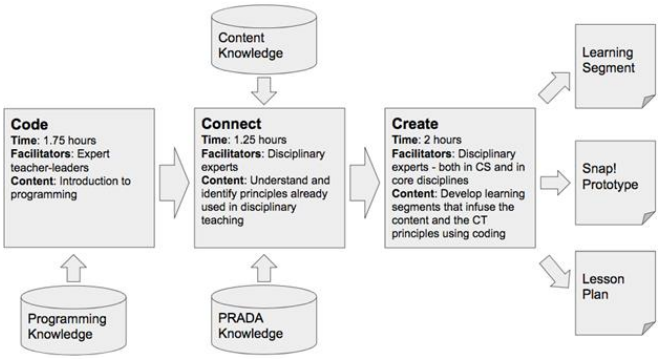

Figure 1: Code, Connect, Create (3C) Professional Development Code

3.1 Code

The goal of the first 3C PD model component, Code, is to provide participants with opportunities to build their understandings of CT principles, learn programming concepts, and gain experience in programming with Snap!. One of the primary affordances that led to the selection of Snap! is its focus on helping users create their own new modules, called custom blocks, by giving a name to a set of blocks and hiding the details of the block. Custom blocks have been used to extend Snap! to build in complex functionalities, such as scientific or mathematical simulations in Netsblox [6]. This affordance is particularly important for teachers needing to customize programming projects for different levels of abstraction or complexity to align with their pedagogical needs and curricula. Further, Snap! can allow students to ignore or explore code as needed. Code sessions also provide teachers with opportunities to collaborate with each other and with facilitators, including expert teacher-leaders and university faculty, to familiarize themselves with the functionality of Snap! as a programming language and to get hands-on experience in coding from the student perspective.

During 105-120 minute Infusing Computing Code sessions held each day, teachers were introduced to computational thinking concepts (e.g., algorithms, abstraction, pattern recognition, decomposition), which were then reinforced in the Connect sessions, as well as programming-specific terms (variables, loops, conditionals, lists). Each session was designed using a Use-Modify-Create structure, where teachers were expected to use existing code, modify code to solve Parson's problems and create code for use in their learning segments. Each programming activity was broken down into five to seven tasks, from reading worked examples, fixing bugs in the code, to creating a short script. Table 1 lists the specific topics covered in the Code sessions, which include repeat loops, variables, mathematical operators, custom blocks, and labels.

3.2 Connect

The goal for the Connect sessions is for teachers to engage with content area colleagues to refine their understandings of CT principles outlined in the PRADA model. Connect session content is designed to reinforce computational thinking concepts and vocabulary introduced during the Code sessions. These daily sessions focus on a particular PRADA theme (pattern recognition, abstraction, decomposition, and algorithms) and begin with a whole-group discussion in which teachers discuss concrete examples of how their existing teaching practices included PRADA elements, even if they did not use the PRADA terminology at the time. To support teachers in identifying integration opportunities and to demonstrate the inherent connections between CT and their disciplinary teaching, teachers examine their standards and brainstorm sample plugged or unplugged activities that would support students' disciplinary understandings, as well as their developing knowledge of CT concepts.

During the 60-minute Infusing Computing Connect sessions, teachers engaged in a collaborative standards mapping activity in which they were asked to identify a standard number and indicator, describe the standard, identify and explain related PRADA elements, and suggest sample plugged or unplugged activities that could be implemented in the classroom setting. Standards mapping was done collaboratively, using Google Forms and a spreadsheet with exported responses so that all teachers could see their colleagues' work to make explicit connections between PRADA elements and standards. Table 2 contains sample participants' standards mapping responses.

3.3 Create

One of the key elements of the 3C model is that it provides opportunities for participants to engage in CT infusion and programming as both learners and teachers. Each Create session begins with a discussion in which teachers reflect on their new learning from the two morning sessions, develop goals for designing and creating their learning segment, and discuss areas of need. Then, participants work in teams or individually to develop their learning segments. Participants also create materials to present their learning segments to other participants, as well as invited guests and school administrators, during a Demo Fair on the final day of the workshop. The Create session time is specifically designed for flexible utilization, as the ways that teachers will use the time will vary based on the nature of their intended learning segment.

During the 120-minute Infusing Computing Create sessions, teachers utilized varying resources and types of support that depended on their needs and the goals of their particular learning segments. For example, to create learning segments that centered around teacher-created simulations that required students to use or modify existing code, participants worked in close collaboration with computer science experts to explore the Snap! Programming language. One middle school science teacher, for instance, designed a Snap! thermometer lab and starter code for a 9th-grade biology course in which students code adjustments to calculate energy output based on different temperatures. This required her to work closely with computer science experts to create custom blocks and to refine the Snap! environment to maximize student learning. Other participants chose to dedicate the majority of their work time to the design of the learning segment plan and pedagogical artifacts. This was particularly important for teachers who designed learning segments that required students to learn coding skills to create original products. For example, one group of middle school teachers created a multi-day learning segment that involved students' original design and programming of quilt squares reflecting Appalachian culture. In order to be able to anticipate students' challenges and to build pedagogical supports

to help them connect to multiple disciplinary standards, this group developed a multi-faceted system of support that included introductory slide shows, reading selections, multimodal materials, and student tutorials.

4 RESULTS AND IMPACT OF THE 3C MODEL

Our analysis of the impact of the 3C model focused on examining changes in teachers' self-efficacy and beliefs regarding CT integration into disciplinary content. Analysis of quantitative survey results revealed significant shifts in teacher self-efficacy. For instance, on the pre-PD surveys, 78% of teachers (n=115) rated their knowledge and understanding of CT infusion as poor or fair, with only 7.6% rating themselves as very good or excellent. By the end of the PD, 48.3% of teachers rated themselves as very good or excellent, with only 16.9% rating their understandings of CT infusion as poor or fair. Further, on the post-PD surveys, more than 90% of teachers (n=114) either agreed or strongly agreed with all three of the following statements: "I am more likely to incorporate CT activities in my classroom, I can more effectively design CT activities, and I can better engage students in making sense of CT and designing solutions to problems".

Further analysis of teacher implementation surveys (n=26) returned after the 2018-2019 academic year revealed that the 3C model extended teachers' knowledge, skills, and performances so that they could implement their new knowledge in the classroom setting. Overall teacher beliefs regarding the impact of the professional development on classroom instruction were measured using 5 Likert scale items, with 1 meaning that they strongly disagreed with the item and 5 meaning that they strongly agreed (see Table 3).

Qualitative responses also indicated that the model was an impactful component of teachers' experience. As a high school, social studies teacher said, "The summer PD was outstanding and I learned so much that I was able to bring back to the classroom. The techniques I learned invigorated my curriculum and challenged my students."

In order to understand changes in teachers' beliefs from pre-PD to post-PD, we also categorized qualitative responses on pre and post-PD survey items targeting their goals for classroom integration of CT into one of five themes: enrichment, review, collaboration, disciplinary-computational thinking connections, and unsure (see Table 4 for definitions and examples of codes used for pre-PD and post-PD responses).

Table 1. Initial Plan for Code Session Activities, Main CT Concepts, and Programming Concepts Introduced or Highlighted

	Monday	Tuesday	Wednesday	Thursday
Activity	Introduction to the Snap programming environment. Draw a square, a triangle, a house, and a row of houses	Create and manipulate a shopping list in Snap	Data exploration and visualization by creating graphs in Snap	Explore the Cellular programming environment and the Epidemic Disease Activity Example
Main CT Concept	Abstraction	Pattern Recognition & Decomposition	Algorithms	All PRADA
Programming Concepts Introduced or Highlighted	Custom blocks, Variables, and Repeat loop	Data Structures- lists as containers for data, conditionals for decision making, Variables for storing input	Abstract Data Types, Global Variables, Labels, Mathematical operators	Event-driven programming, Forever loops, relational operators, Manipulating Variables

Table 2. Sample Standards Mapping Responses

Grade and Subject Area	Standard	PRADA Connections	Sample Activity	Plugged or Unplugged
9th Grade Algebra	Solve quadratic equations in one variable that have complex solutions.	Abstraction, Algorithm Students will use the quadratic formula, which will require following an algorithm to solve the problem correctly	Students will break down the process step by step, determine the possible outcomes (discriminant), and create the SNAP code to apply the quadratic formula to any quadratic equation	Both
7th Grade World History	Summarize mercantilism as a way of building a nation's wealth, including government policies to control trade.	Abstraction Standard calls for summarizing while abstraction focuses on hiding details	Analyze the image and create a chart with the main points. Drag and Drop Items from colonies to benefit mother countries	Both
9th Grade Biology	Develop and use models to exemplify the changes that occur in a cell during the cell cycle and predict, based on the models, what might happen to a cell that does not progress through the cycle correctly.	Decomposition Students must decompose all components of the cell membrane and how they interact to pass materials through it.	Building a Cell Membrane Challenge (limited materials must be "purchased", used in the construction, and demonstrate passive and active transport).	Both

Then, to triangulate responses across data sources, we analyzed the teacher-created CT-integrated lesson plans, as well as reflective surveys from the implementation of the lessons during the 2018-2019 academic year.

Before the PD began, teachers viewed CT as either enrichment that went above and beyond the borders of their curricula, or as a way to foster student engagement and collaboration. Specifically, on the pre-PD survey, the majority of teachers (51%) described their goals for integrating computing as providing enrichment beyond the existing curriculum for a small group or whole class of students (45%) or reviewing content or material already covered in class (6%). While 24% noted that they hoped to be able to integrate or infuse computational thinking into their disciplinary teaching and 9% referenced interdisciplinary collaboration, there was a lot of uncertainty regarding how these goals might be accomplished. For instance, one teacher noted that her goal was "making connections between math to science," while another said, "I hope to teach my students how to use coding to perform mathematical computations that we would be using in my content areas of Math," but details about how they could support these connections were missing entirely. Additionally, 16% were unsure of how they would integrate computing into the classroom; as one teacher said, she hoped to "make my students do some projects with coding," but thought that she "should learn more before I start using coding in my classroom."

As teachers progressed throughout the workshop, many began to see the infusion of CT into their content teaching as the primary goal of the professional development. On the post-PD survey, 70% of teachers reported that the primary goal of their learning segments was to integrate CT into their disciplinary teaching. As one teacher noted, "Our project is taking an idea from marine science and giving the students an opportunity to learn coding in order to develop a model to demonstrate their learning." Further, 4% reported that the primary goal was collaboration and 7% reported a primary goal of reviewing concepts covered in current or previous classes. However, within the review category, several teachers referenced the importance of review in solidifying learning; for example, as one teacher said, "We made modest gains in the ACT composite score last year. We will continue to address strengthening language, math, science, and writing. Implementing a novel forum for improvement will ensure that learning isn't forgotten with the conclusion of the testing administration." Finally, 17% of teachers reported conceptualizing their learning segments as enrichment projects that would supplement the existing curriculum, which represented a 28% decrease from the beginning of the week.

Analysis of implementation data, including lesson plans, teacher reflections, and video recordings of teacher lessons, showed that some teachers were able to successfully embed computational thinking into disciplinary content. A group of middle school science teachers, for example, collaboratively designed a lesson in which students explored the structure of water molecules to reinforce their knowledge of the properties of water. Kevin, who implemented the lesson in Spring 2019, said, "The end result of the lesson is that each student group was able to draw the water molecule and create a program of molecule coming together." Kevin did note several changes that he would make to the lesson implementation, stating, "When I implement Snap! next year, I will develop lessons that sequentially build programming skills. For example, lessons will start by building simple skills and culminate by integrating all the skills."

Table 3. Teacher Beliefs Regarding the Impact of the 3C Model

Likert Items	Mean (n=26)
Infusing Computing extended my knowledge, skills, and performances.	4.73
What I learned during the Infusing Computing program positively impacted the achievement of my students.	4.19
The content of Infusing Computing is relevant to my professional responsibilities.	4.38

Table 4. Themes, Definitions, and Examples of Codes for Teachers' CT Infusion Goals

Theme	Definition	Example
Enrichment	Activities, materials, or resources that would be utilized as an addition to the existing curriculum for individual students, small groups, or the whole class	"I will try to integrate it as a differentiation piece for small collaborative groups or individual instruction." (pre-PD)
Review	Review of concepts, ideas, or content covered in current or previous courses	"Hopefully, the app will allow my students to practice for the AP exam" (post-PD)
Collaboration	Working with other teachers and colleagues to develop collaborative materials and activities	"I hope to collaborate with others so that I can create transdisciplinary STEM units that will help foster critical thinking skills to help my students become geared for career readiness." (post-PD)
Disciplinary-Computational Thinking Connections	Integrating CT into disciplinary teaching to support students' development of content knowledge, CT knowledge, or both	"As a science teacher, I plan to develop lessons for students to create models and simulations related to chemistry (building molecules and atoms), develop water quality assessment programs related to hydrology, and have students develop programs to collect data/surveys related to biotechnology practices." (pre-PD)
Unsure	Uncertainty regarding the goals of the workshop or ideas for CT infusion	"I'm getting my bearings as it pertains to coding and I feel like I need to let it all sink in before I can begin the integration phase." (pre-PD)

While many teachers were able to successfully implement their lessons, others faced several barriers, including a lack of technological resources, difficulty in aligning schedules to conduct interdisciplinary lessons, and the need to more fully develop a series of integrated lessons for ongoing implementation. As one high school algebra teacher said, "I feel that I need to expand upon it to make the coding practices more successful but in the limited time we spent on coding, I think my students made great progress. I'd like to add additional coding lessons to each of the Algebra units to reinforce how simple algorithms (like solving equations) can be automated by the computer." As we discuss in the next section, we are currently working on designing systems of ongoing support, including student tutorials and technical guides, in order to help teachers overcome some of these barriers.

6 CONCLUSION

Given the fundamental role of computational thinking to nearly all elements of 21st-century life, there is a growing need to develop systems that enable P-12 teachers to fully integrate CT within existing curricula and disciplinary teaching. It's essential that teachers come to understand the value of computational thinking--not just as an isolated concept that relates to computer science, but also as a way to enhance and support more complex discipline-specific and interdisciplinary understandings. Our findings offer important implications for increasing access to computational thinking through supporting teachers in integrating CT in middle and high school content classrooms. Analysis of data from the implementation of the 3C model indicates that it effectively supported shifts in teacher beliefs and self-efficacy regarding the role of CT in P-12 classrooms. Further, findings suggest that teachers found great value in learning to code, thinking critically about connections between disciplinary content and computational thinking principles, and collaborating with colleagues to support discipline-specific and interdisciplinary learning. Overall, we believe that the combination of Code, Connect, and Create sessions helped to shift teacher beliefs about the role of CT in disciplinary content, while simultaneously addressing their concerns that they might be unequipped with the programming experience and knowledge to support students in learning to code.

While the 3C model proposes a new framework that has promise for supporting teachers in learning to integrate computational thinking into their classrooms, there are still many areas that need further exploration and analysis. First, there is a pressing need to develop ongoing, flexible, and differentiated scaffolds to support teachers' computational thinking infusion. A key next step is for researchers to collaborate with practitioners to study the impact of tiered supports that can prevent teachers' knowledge from becoming inert [30]. Further, in order to support sustained integration into disciplinary content and curricula, studies must identify explore the forms and functions of collaborative, school and district-based communities for teachers to reflect on pedagogical innovations and failures.

In order to address teacher needs that we identified during our analysis of Infusing Computing teacher implementation data, we are currently engaging in design research to develop, evaluate, and refine scaffolds to extend and support ongoing teacher learning during the academic year implementation. Examples of supports that we are currently analyzing include: 1) monthly webinars, with topics ranging from unplugged CT lessons to assessment of CT; 2) using Slack as a community-building tool; 3) podcasts as a way to support asynchronous engagement; and 4) the development of repositories for sharing lesson plans, resources, and feedback from classroom infusion experiences. Further, during the Summer 2019 Infusing Computing workshop, we introduced and are currently analyzing the impact of differentiated 3C sessions, including targeted Code sessions using Python and Connect sessions for teachers in instructional leadership roles within their schools. It is our hope that engaging in these cycles of design and analysis will allow us to establish a comprehensive model of teacher professional development that will not only increase access to CT but can also promote more critical connections to content area learning and interdisciplinary thinking.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant numbers 1742351 and 1742332.

REFERENCES

[1] Dan Garcia, Brian Harvey, and Tiffany Barnes. 2015. The beauty and joy of computing. *ACM Inroads*, *6*, 4 (2015), 71-79.

[2] Yihuan Dong, Veronica Cateté, Robin Jocius, Nicholas Lytle, Tiffany Barnes, Jennifer Albert, Deepti Joshi, Richard Robinson, and Ashley Andrews. 2019. PRADA: A practical model for integrating computational thinking in K-12 education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. ACM, New York, NY, 906-912. DOI: 10.1145/3287324.3287431

[3] Charoula Angeli, Joke Voogt, Andrew Fluck, Mary Webb, Margaret Cox, Joyce Malyn-Smith, and Jason Zagami. 2016. A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Journal of Educational Technology & Society* 19, 3 (2016), 47-57.

[4] 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 (February 2011), 48-54.

[5] Hilda Borko. 2004. Professional development and teacher learning: Mapping the terrain. *Educational Researcher*, 33, 8 (2004), 3-15.

[6] Brian Broll, Akos Lédeczi, Peter Volgyesi, Janos Sallai, Miklos Maroti, Alexia Carrillo, Stephanie L. Weeden-Wright, Chris Vanags, Joshua D. Swartz, and Melvin Lu. 2017. A visual programming environment for learning distributed programming. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, 81-86.

[7] Kathy Charmaz. 2006. *Constructing grounded theory*. Sage, London, UK.

[8] Code.org and CSTA. 2018. 2018 State of Computer Science Education. Retrieved from https://advocacy.code.org/

[9] David Cohen, Stephen W. Raudenbush, and Deborah Loewenberg Ball. 2003. Resources, instruction, and research. *Educational Evaluation and Policy Analysis* 25, 2 (2003), 119-142.

[10] Barney Glaser and Anselm Strauss. 1999. *Discovery of grounded theory: Strategies for qualitative research*. Routledge, London, UK.

[11] Joanna Goode and Jane Margolis. 2011. Exploring computer science: A case study of school reform. *ACM Transactions on Computing Education (TOCE)* 11, 2 (2011), 12-16.

[12] Google. 2018. What is computational thinking? Retrieved from https://computationalthinkingcourse.withgoogle.com

[13] Shuchi Grover and Roy Pea. 2013. Computational thinking in K–12: A review of the state of the field. *Educational Researcher* 42, 1 (2003), 38-43.

[14] Brian Harvey and Jens Mönig. 2010. Bringing "no ceiling" to Scratch: Can one language serve kids and computer scientists? *Constructionism* (2010), 1–10.

[15] Yasmin Kafai and Quinn Burke. 2013. Computer programming goes back to school. *Phi Delta Kappan* 95, 1 (2013), 61-65.

[16] 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.

[17] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 1-15.

[18] Punya Mishra, Matthew J. Koehler, and Danah Henriksen. 2011. The seven trans-disciplinary habits of mind: Extending the TPACK framework towards 21st century learning. *Educational Technology* 11, 2 (2011), 22–28.

[19] National Research Council. 2012. *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas*. National Academies Press, Washington, DC.

[20] Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas.* Basic Books, New York, NY.
[21] Pratim Sengupta, John S. Kinnebrew, Satabdi Basu, Gautam Biswas, and Douglas Clark. 2013. Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies* 18, 2 (2013), 351-380.
[22] Lee S. Shulman and Judith H. Shulman. (2004). How and what teachers learn: A shifting perspective. *Journal of Curriculum Studies* 36, 2 (2004), 257-271.
[23] Valerie Shute, Chen Sun, and Jodi Asbell-Clarke. 2017. Demystifying computational thinking. *Educational Research Review* 22 (2017), 142-158.
[24] Anselm Strauss and Juliet M. Corbin. 1990. Basics of qualitative research: Grounded theory procedures and techniques. Sage, London, UK.
[25] David Weintrop, Elhelm 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.
[26] Suzanne Wilson and Jennifer M. Berne. (1999). Teacher learning and the acquisition of professional knowledge: An examination of research on contemporary professional development. *Review of Research in Education* 24, 1 (1999), 173-209.
[27] Jeannette M. Wing. 2006. Computational thinking. *Communications of the ACM*, 49, 3 (2006), 33-35.
[28] Ursula Wolz, Meredith Stone, Kim Pearson, Sarah Monisha Pulimood, and Mary Switzer. 2011. Computational thinking and expository writing in the middle school: A novel approach to broadening participation in computing. *ACM Transactions on Computing Education* (TOCE) 11, 2 (2011), 61-83.
[29] 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 (2016), 565-568.
[30] Aman Yadav, Chris Mayfield, Ninger Zhou, Suzanne Hambrusch, and John T. Korb. 2014. Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education* (TOCE) 14, 1 (2014), 1-16.
[31] Aman Yadav, Ninger Zhou, Chris Mayfield, Suzanne Hambrusch, and John T. Korb. 2011. Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCS '11).* ACM, New York, NY, 465-470.