PRADA: A Practical Model for Integrating Computational Thinking in K-12 Education

Yihuan Dong[1], Veronica Cateté[1], Robin Jocius[2], Nicholas Lytle[1], Tiffany Barnes[1], Jennifer Albert[2], Deepti Joshi[2], Richard Robinson[2], Ashley Andrews[2]
[1]North Carolina State University, Raleigh, North Carolina
[2]The Citadel, Charleston, South Carolina
{ydong2,vmcatete,nalytle,tmbarnes}@ncsu.edu
{rjocius,jalbert,djoshi,rjmr,ashley.andrews}@citadel.edu

ABSTRACT

One way to increase access to education on computing is to integrate computational thinking (CT) into K12 disciplinary courses. However, this challenges teachers to both learn CT and decide how to best integrate CT into their classes. In this position paper, we present PRADA, an acronym for Pattern Recognition, Abstraction, Decomposition, and Algorithms, as a practical and understandable way of introducing the core ideas of CT to non-computing teachers. We piloted the PRADA model in two, separate, week-long professional development workshops designed for in-service middle and high school teachers and found that the PRADA model supported teachers in making connections between CT and their current course material. Initial findings, which emerged from the analysis of teacher-created learning materials, survey responses, and focus group interviews, indicate that the PRADA model supported core content teachers in successfully infusing CT into their existing curricula and increased their self-efficacy in CT integration.

CCS CONCEPTS

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

KEYWORDS

Professional Development; Computational Thinking

ACM Reference Format:
Yihuan Dong, Veronica CatetÃĺ, Robin Jocius, Nicholas Lytle, Tiffany Barnes, Jennifer Albert, Deepti Joshi, Richard Robinson, 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), Feb. 27–Mar. 2, 2019, Minneapolis, MN, USA. ACM, NY, NY. 7 pages. https://doi.org/10.1145/3287324.3287431

SIGCSE '19, February 27-March 2, 2019, Minneapolis, MN, USA

ACM ISBN 978-1-4503-5890-3/19/02. . . $15.00
https://doi.org/10.1145/3287324.3287431

1 INTRODUCTION

Computational Thinking (CT) is the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms [1], which often include problem decomposition and abstract reasoning. As computational technologies are becoming a more integral part of our lives and workforce, it is important for students to develop these CT skills early and with equal opportunity in order to prepare them for their future careers. Teaching CT in core classes has the advantage to avert the selection bias present in elective or after school programs [21]. However, this challenges us to effectively train teachers to be able to understand CT concepts, identify CT components in their lessons, and foster a CT mindset among their students.

After Wing re-introduced the concept of CT in 2006 [25], a number of attempts have been made to define what CT is at various granularities [2, 9, 22, 24]. However, many definitions include Computer Science terms (automation, parallelization, etc.) that are not only hard to explain in any K-12 classroom, but are also difficult to understand without actual "coding" activities, which not all teachers have the resources or knowledge to facilitate. Furthermore, CT professional development (PD) without intentional content contextualization fails to offer affordances for how teachers can utilize CT to support disciplinary learning goals. There is a need for a CT definition that teachers can effectively use to communicate CT to students in core classroom settings. Such a CT definition needs to be:

(1) understandable by K-12 teachers, the majority of whom have limited Computer Science knowledge
(2) integratable into existing curricula, and
(3) generalizable to any discipline

In this paper, we present PRADA (an acronym for Pattern Recognition, Abstraction, Decomposition, and Algorithms) as a practical and understandable way of introducing the core ideas of CT to non-computing teachers in order to support them in infusing CT into their curricula. We piloted PRADA as part of a summer PD on infusing CT into middle and high school content area courses. During the PD, we taught and encouraged teachers to both recognize the PRADA elements that already exist in their courses, and also helped them to create digital artifacts in a block-based programming language to facilitate teaching PRADA. We approached the professional development with the following preliminary questions:

(1) Were the teachers able to create computational artifacts?
(2) Were the teachers able to correctly identify the PRADA components in their artifacts?

(3) Did teachers show an increase in self-efficacy towards understanding and teaching PRADA?
(4) Were there any interesting trends in teacher created artifacts and feedback?

This paper presents our preliminary results which emerged from interpretations of daily teacher feedback, semi-structured interviews, and summative PD surveys, but focused primarily on the teachers' products: lesson plans and block-based programs. We begin with an overview of CT frameworks and pedagogical strategies for CT PD. Then, we make the case for the use of PRADA as an understandable way of introducing CT to non-computing K-12 teachers. The next part presents the methodology used for this investigation and continues on to show evidence from the teacher artifacts indicating exemplars of successful implementa- tions of PRADA. We conclude with a discussion of our findings and recommendations for future CT teacher professional development.

2 LITERATURE REVIEW

In the last decade, the concept of Computational Thinking has undergone several reformations. Wing has moved CT from the concepts fundamental to computer science (solving problems, designing systems and understanding human behavior [25]), to the "thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" [24]. The important distinction here is that the solutions can be carried out by another agent effectively.

In 2011, the International Society for Technology in Education (ISTE) and Computer Science Teachers Association (CSTA) came together to create an operational definition of CT that could provide a framework and set of vocabulary terms for all K-12 teachers [9]. They describe CT as a problem-solving process that includes formulating problems, logically organizing data, representing data through abstractions, automating solutions, reflecting on the efficiencies of possible solutions, and generalizing and transferring this process to a variety of problems. However, while these ideas are general enough to apply to any discipline, there remains a lack of clarity with regard to how teachers can use CT to support content learning and a need for more concrete models and supports for classroom integration.

Several attempts have been made to map CT specifically to other subject areas. Weintrop et al. have developed a taxonomy of CT practices for both Math and Science disciplines drawing on major connections to modeling and abstraction [21, 22]. Each of the four CT skill sets identified by Weintrop and his team evolved from reviews of literature and common coursework, as well as interviews and discussions with K-12 educators and researchers. Both Data and Information skills, as well as Modeling and Simulation skills, map directly to STEM practices. The new skills, Computational Problem Solving and Systems Thinking, focus more directly on practices deliberately expressed in Computer Science, such as debugging, assessment of solution effectiveness, and thinking in levels, which might be difficult for K-12 teachers to easily recognize within their own curricula or unpack for students.

Similarly, the vocabulary terms used in the early CT framework identified by the National Research Council [4, 7] present definitional challenges for teachers. In this framework, in addition to data manipulation and simulations, both parallelization and automation are used as high level concepts. Although these last two skills are useful for software engineers, these may be difficult or impractical to integrate into middle and high school courses.

In order to demystify the terminology around CT [4], a simplified skill set and vocabulary has been defined by k12cs.org and further modified by Google Education [10, 16]. K12cs.org focuses on: Defining problems; developing and using Abstractions; Creating computational artifacts; and Testing and refining artifacts. Although simplified, not all classrooms have ready access to computing technology, so the latter half of the CT skills can be difficult for teachers to implement. Instead, Google focuses on Problem decomposition, Pattern recognition, Abstraction, and Algorithm design [10]. As stand-alone or intermixed concepts, these can be implemented into other subjects by simply modifying existing lesson plans instead of needing to create something entirely new. An example of each concept is listed below [4]:

Pattern Recognition in Social Studies - identify trends in data from historical or social statistics
Abstraction in Science - simplify models of Newtonian mechanics or solar systems
Decomposition in English/Language Arts - write outlines, identify arguments
Algorithms in Math - list steps for doing long division or integral calculus

Although the historical trend in CT literature has been on identifying key concepts and a core definition [12], there is a new trend exploring how to teach these concepts to in-service teachers [5, 19, 26]. As noted by Margolis [17] and Yadav [26, 27], teachers will not be able to provide their students with quality CT instruction until the teachers themselves understand the construct and feel confident about its integration in their curriculum. If teachers do not feel confident in their abilities to teach computational thinking, students may have negative experiences in learning the concept [13].

Teachers' thinking and practice are understood to be shaped in large part by continued professional learning [3, 8]. Recent research by Bower et al. indicates that "teacher's computational thinking capabilities are relatively malleable" and professional development may sufficiently impact teachers' understanding of CT. According to Bower's study, teachers can learn CT concepts in a short amount of time, building self-efficacy [5]. Furthermore, a 2018 survey by Sands et al. found that there were no differences on teachers' conceptions of computational thinking based upon either the content area (STEM vs. non-STEM) or grade level (primary vs. secondary) [19], making CT PD a level learning field for in-service teachers.

However, there were some limitations to Bower's study: CT was taught in a stand-alone context not situated in the teachers' own domains. While teachers' self-efficacy and conceptual understandings of CT increased after the PD, teachers expressed concern over how to integrate CT into their classrooms [5]. Sands' study suggests that in order to successfully teach CT, training needs to be content-specific and focused on how to integrate computational thinking ideas into existing curricula. Specifically, teachers need

to be introduced to computational thinking in a way that meets their disciplinary learning goals and fits within their pedagogical practices [19] rather than being an "instructional add-on in the K-12 curriculum" [11].

3 DEFINING CT AND PRADA

3.1 What is PRADA

Google's "Computational Thinking for Educators" course identifies Decomposition, Pattern Recognition, Abstraction, and Algorithm design as the key elements that make up Computational Thinking [10]. Even though CT is essential to programming, it can be used to facilitate problem solving across all disciplines, not strictly Computer Science. There is no strict ordering of these elements, and each element can be taught separately. We reordered these elements into PRADA as a mnemonic device to help teachers and students remember the terms. We also revised Google's CT definition, as shown in italics, to make it more adaptable to different disciplines:

Pattern Recognition observing and identifying patterns, trends, and regularities in data, *processes, or problems*

Abstraction identifying the general principles *and properties that are important and relevant to the problem*

Decomposition breaking down data, processes, or problems into *meaningful* smaller, manageable parts

Algorithms developing step by step instructions for solving [a problem] and similar problems

3.2 Why PRADA

There are several advantages in using PRADA as a way of introducing CT. The four elements are detailed enough to capture the essence of CT defined in previous frameworks. We emphasize PRADA as a mindset, not bounded by content area or tools, that helps people solve problems in a systematic and generalizable way. We consider coding to be an important activity that can facilitate the understanding of PRADA, but teaching PRADA does not require teachers to create coding activities or artifacts for their students. PRADA can also be taught with various unplugged activities. Figure 1 shows a mapping of PRADA compared to other popular CT components. PRADA strips out the Computer Science/coding specific components described elsewhere and retains the parts that can be easily generalized and integrated into different disciplines within K-12. Non-CS/coding specific components that are left out, namely Data & Information and Modeling & Simulations, can still be explained with PRADA, as analyzing data generally involves abstracting useful information from recognized patterns, and simulation generally involves decomposing a phenomenon into parts/abstractions, and creating algorithms to mimic observed or hypothesized processes. Additionally, PRADA maintains a meaningful difference between elements such that each element in the PRADA model represents a distinguishable process during problem-solving and can be taught and explained in separate class activities.

PRADA also gives teachers a set of memorable keywords to convey to their students when teaching CT. A previous research study that involved implementing CT in middle school science classrooms shows that the teachers had trouble mapping CT concepts to their teaching material. As a result, the teachers requested a concrete set of names of the key concepts so they could teach and reinforce the terms in each day's activity [6]. PRADA serves as a manageable and memorizable set of keywords that teachers can explicitly introduce and reinforce in their daily classes.

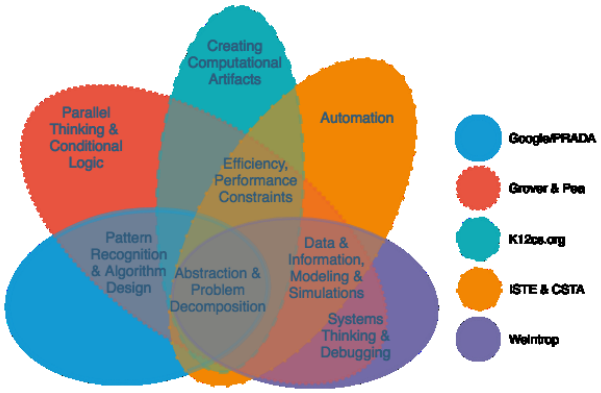

Figure 1: A mapping of popular CT components

In this paper, we argue that the PRADA model can support K-12 teachers' exploration of how to integrate CT into disciplinary classrooms. Our goal is to help teachers realize that many of the pedagogical practices and materials that they already use within their classrooms can address or provide distinct affordances for integrating the PRADA model. However, to do this, teachers need practice and support to recognize and apply CT elements to their own disciplinary content (see examples in section 2). It is with this in mind that we organized our Infusing Computing professional development program to provide opportunities for teachers to work together both within and across disciplines and to learn to create programs with extensive support from computer scientists.

4 METHODS

During two intensive, 5-day Infusing Computing PD workshops in North and South Carolina in Summer 2018, we engaged 116 middle and high school teachers in designing plans to integrate computational thinking into their classrooms. We designed each day of the workshop to include: Snap*!* programming bootcamps, content area focused PRADA sessions, and collaborative development sessions, which culminated in the development of a learning segment that integrates CT into each teacher's disciplinary classroom instruction. Specifically, the PRADA session began with a whole-group discussion of the PRADA element of the day in which teachers discussed concrete examples of how their existing teaching practices included PRADA elements, followed by examining and analyzing, in small groups, the sample classroom artifacts created by the research group. In the development sessions, teachers worked collaboratively in either school or content teams to make their projects. Although teachers were encouraged to create learning segments that allowed students to practice coding, this was not a requirement. Instead, teachers had the flexibility to build lessons around CT using the resources available in their classrooms. Typical learning segment components included a Snap*!*-like prototype, a learning plan that explicitly identified how PRADA elements were

addressed, and supplemental pedagogical materials (e.g., graphic organizers, templates, slideshow presentations, coding guides).

We recruited Math, Science, Social Studies, and English/Language Arts (ELA) content area teachers for grades 6-12 in both states to attend individually or to bring teams from their schools. There were 58 teachers in each of our two workshops, PD1 and PD2, with teachers from several content areas as shown in Table 1. Although many teachers attended in teams, teachers could choose to work individually or with other self-selected teams within the workshop to create learning segments that would be meaningful in their own classrooms. Overall, there were 116 people in 40 teams across the two weeks, with teams ranging in size from one to as many as 6 teachers.

Table 1: Teacher participants and project by discipline

	Math	Science	Humanities	Interdisciplinary
PD1 Teachers	21	24	13	
PD1 # Projects	4	11	4	2
PD2 Teachers	20	19	17	
PD2 # Projects	7	7	3	2

PD sessions were led by members of the research team with support from local area teacher leaders with experience in integrating computing into their curricula. Taking on the roles of participant-observers [23] gave members of the research team the opportunity to interact with participants on a regular basis and to work systematically to understand the teachers' developing understandings of the PRADA model, as well as their beliefs and expectations regarding integrating CT into their classrooms. We believe that these roles allowed us important insights into the data analyzed in this paper, particularly given that the goal of this work is the evaluation of teachers' understanding and implementation of PRADA, rather than their perceptions of the PD or facilitators.

4.1 Data Collection & Analysis

Primary sources of data for this analysis included teacher products (block-based programming prototypes, learning segment plans, and related artifacts), pre- and post-PD surveys, daily reflections, feedback forms for teacher presentations, and audio-recorded teacher interviews. Data analysis proceeded in three overlapping and recursive phases: systematic review and coding of learning segment artifacts, purposive sampling and analysis of focal learning segments, and triangulation of data sources.

During the first phase of analysis, the first author, an expert who participated in the design and the implementation of the PD, reviewed the learning segments and the block-based programming project files of all the collaborative teacher submissions (n = 40). While many of the products have interdisciplinary connections, of the 40 products, 11 were primarily designed for math, 18 for Science, 7 for Humanities (Social Studies and English Language Arts), and 4 were purposefully designed for all disciplines. Products included a form where teachers were to elaborate how their planned lessons would target each PRADA concept. Each submission was reviewed to determine whether or not the interpretation of PRADA in the context of the lesson plan agreed with what we intended the teachers to learn. Based on our expert review, most teams successfully implemented PRADA as 32.5% of the projects had a correct explanation and implementation for all 4 of the PRADA elements, and 32.5% of the projects had a correct explanation and implementation for two or more elements of PRADA. Of the remaining 13 projects, 6 identified PRADA elements by name, 4 incorrectly identified PRADA elements, and 3 made no mention of PRADA.

In the second phase, we used purposive sampling to select four case studies representing math, science, humanities, and interdisciplinary products to analyze evidence of teachers' understandings of the PRADA model within their learning segments. Purposive sampling is "based on the assumption that one wants to discover, understand, or gain insight; therefore one needs to select a sample from which one can learn the most" [18]. The case studies selected were based on the following criteria: the lesson plan is complete, the learning segment or supporting document clearly describes how each of the four elements of PRADA is identified and used in the lessons, and the block-based coding prototypes were completed with a clear indication of the presence of some PRADA elements (e.g. proper use of the repeat block demonstrating understanding of pattern recognition).

During the third and final phase of analysis, we focus on participant feedback. The project team examined feedback from presentations, survey results, and teacher interviews to triangulate findings across sources using the constant comparative method [20].

5 CASE STUDIES

We report our preliminary findings from the PRADA themed Infusing Computing PD using four exemplary projects. We intend these case studies to demonstrate how teachers successfully identified and infused CT into their own domains, as well as to highlight how teachers were able to bridge domains using PRADA. After we present a description of each product, we list the teacher explanations for integrating PRADA in quotes, and we offer our own interpretations of the product's relation to each PRADA element in italics. We also discuss how these exemplars illustrate opportunities for additional support that may be needed for teachers struggling with some elements of CT.

5.1 Interdisciplinary Exemplar: Life in the Middle Ages

One interdisciplinary group of high school teachers made a historical puzzle game for students to explore life in the Middle Ages through puzzles and quizzes. Table 2 shows how the teachers explained the PRADA elements, providing good examples and activities in all cases except for abstraction.

5.2 English/Language Arts Exemplar: Historical Game

In our second exemplar for 6th grade English Language Arts, teachers created a historical role-playing game to provide context for students' reading of narrative and expository texts. Table 3 shows how this team was able to identify and apply all four PRADA elements in their learning segment and identified student learning for patterns, abstraction, and decomposition.

Table 2: PRADA in the interdisciplinary product

Pattern Rec.: "Students should identify the repetition in games, the progression of the timeline, characters, puzzle patterns, and the patterns in graph plotting".
Abstraction: Students would demonstrate abstraction through "overarching knowledge of the middle ages." *This suggests that the teachers either consider the knowledge to be abstractions, or they did not understand abstraction enough to provide concrete examples.*
Decomposition: Students should "research events and note dates in the timeline, discover specific characteristics from the prologue of the game, break down the logical components in the puzzle games, and so forth".
Algorithms: Students would design algorithms to "code the different parts of the game and create a storyboard".

Table 3: PRADA in the humanities product

Pattern Rec.: During their explorations of the world, the teachers task students with recognizing the "repetition in history, storytelling, coding" when clicking on similar items.
Abstraction: "Understand history inspiring fiction" *This demonstrates a good disciplinary use of the concept of abstraction, drawing parallels between history and stories.*
Decomposition: Students should "break down plot diagrams into subcategories and look further into what blocks go into a specific abstraction". *This implies a good understanding of our CT definitions of decomposition and abstraction (naming a functional group of blocks) and the attempt to connect decomposition to abstraction.*
Algorithms: In their Snap*!* file, teachers used multiple sprites to handle different functionalities, reused code in different sprite objects, and correctly coordinated the logic and timing between the sprites, demonstrating impressive understanding. *Teachers did not specify student activities for algorithms.*

5.3 Science Exemplar: Rock & Mineral Quiz

The science exemplar, Rock & Mineral Quiz, was developed for students in 6th - 10th grade science classes to review and practice knowledge learned about the components of rocks and minerals in rock cycles. The first Snap*!* project created by the teachers was designed to monitor and evaluate students' ability to respond to questions related to rocks and minerals throughout a class period. The other project is intended to be used in a exploratory lab activity, where the students measure different properties of rock specimens and input the measurements into the Snap*!* project to check whether they made correct inferences. The project files demonstrate a clear understanding of pattern recognition and decomposition through the reuse of code segments and use of multiple sprites and control structures to handle different logic. There is no clear indicator of abstraction in either of the project files. Table 4 shows the break-down of PRADA elements listed for this group. We felt that the listed elements for Decomposition and Algorithms may need to be reversed, but they could be appropriate depending on how they are carried out and discussed in classrooms.

Table 4: PRADA in the science product

Pattern Rec.: Students should "recognize the composition of rocks throughout the rock cycle" (science) and "recognize patterns in the coding language and how to incorporate loops" (coding).
Abstraction: Students should identify global and local properties (science) and code them as global and local variables in Snap*!* (coding).
Decomposition: Students should "break down the composition of minerals by characterizing density, hardness, and strength" (science) and "will complete sorting activities to learn where command blocks are located" (coding).
Algorithms: Students investigate "the rock cycle in a series of labs and activities" (science) and "break down the previously written code for a review game in Snap*!* by replacing lines of code with different questions" (coding).

5.4 Math Exemplar: ACT Football

Our exemplar math product is an ACT practice quiz football game. Each question is associated with a number of yards that will advance their team if answered correctly, or penalize their team if answered incorrectly. The teachers described two ways of using the game: as a practice for the ACT test and as a way of having the students learn CT and coding skills. Table 5 demonstrates the use of PRADA by this team.

Table 5: PRADA in the math product

Pattern Rec.: Students will be "questioned on pattern recognition as they play, such as:'what does the game do when a correct answer is given? A wrong answer?'" They expect the students to be able to notice the repetitive patterns in the game and familiarize themselves with the game mechanics.
Abstraction: Teachers will "display a [custom] block and explain [how it is] considered an abstraction" that is "composed of an algorithm".
Decomposition: In the Snap! project, teachers demonstrated understanding of decomposition by creating custom blocks named "coin toss", "initial possession", and "play" *Teachers did not specify how students would practice or demonstrate decomposition.*
Algorithms: Students will "use an algorithm to decide what level of questions they want to answer which will determine their advancement in the football game". Later, the students will be asked to "use an algorithm to create a block of code that will ask a question and respond to the answer as to whether correct or incorrect", which is an algorithm the teachers crafted in the "play" block in their Snap! project.

6 ANALYSIS OF TEACHER FEEDBACK

The analysis of teacher post-PD surveys demonstrates tha PRADA was warmly received. In both PDs, respondents showed a high level of willingness to adopt PRADA and high self-efficacy in being able to design and teach using PRADA. Table 6 shows the average response for Likert-style questions participants received.

Table 6: Average rating for teacher Self-Efficacy (1 strongly disagree to 5 strongly agree)

	PD1	PD2
I am more likely to incorporate PRADA activities in my classroom.	4.59	4.55
I can more effectively design PRADA activities.	4.43	4.40
I can better engage students in making sense of PRADA and designing solutions to problems.	4.46	4.40

Our analysis of teachers' daily reflections and interviews reveals that PRADA seemed to be well-received because its elements connected easily with already established concepts in the teachers' disciplines. For example, as one teacher notes, "Abstraction for math is basically modeling - I do it every day!". As they established connections between CT and existing curricula, teachers began to understand that they already engage in computational thinking ("in math, we use abstractions all the time"), but they now had the toolkit to be able to express these ideas using CT language. As one teacher put it, "The introduction of the terminology of PRADA was helpful, as teachers realized that they are already doing this." Another said, "... I really do this within my teaching already. I just need to share with students this terminology in science class."

Teachers were also able to identify CT in their colleagues' learning segments. For instance, as one teacher noted in their feedback to a colleague, "I love that [students] can do a creative model of their physical quilt that models computational thinking." Another teacher focused specifically on the disciplinary elements of calculus that mirrored PRADA: "It is interesting that calculus in and of itself is representative of computational thinking. It's very "meta" that the students are creating a program to better understand multiple representations while they are simultaneously gaining a deeper understanding of PRADA concepts that are mirrored in calculus concepts."

However, while teachers had success in both identifying and creating connections with PRADA elements, some confusion existed in how to teach these elements within a classroom. As one teacher put it: "HOW do you teach abstraction? I can recognize it, but what are ways that I can emphasize it, and have my students value abstraction?" This was further compounded with some teachers still having confusion over to what degree they needed to incorporate programming into their teaching of CT ("Are we expected to have the students code, or, is that just an option?").

A few teacher products did not explicitly relate PRADA elements to their disciplines, but only indicated how PRADA relates to coding. This may reflect a potential lack of understanding of PRADA as a connector between CT and disciplines; as one teacher said, "PRADA helps me work in coding in my classroom". However, this may also simply be a limitation of our data collection instrument which only provided small text boxes for explaining how their products related to PRADA. In our 3C model of Code, Connect, Create [15] for integrated CT professional development, teachers attended sessions on learning to code, to connect PRADA to their disciplines, and to create their learning segments for the classroom. Teachers may have been eager to demonstrate their understanding of PRADA in the context of coding, rather than in their own disciplines.

7 DISCUSSION & CONCLUSIONS

Unlike other coding-focused CT models, PRADA is an abstraction of the mindset that is commonly used by computer scientists and programmers when solving problems. We emphasize the word mindset because PRADA is a thinking process, not bounded by content area or tools, to help people solve problems in a systematic and generalizable way. When helping teachers make connections to their disciplinary areas, we have found that while STEM areas have natural connections to CT through activities like modeling, simulations and experimental methods (e.g., NGSS), other areas like English Language Arts can also benefit from applying and recognizing CT. However, before a teacher can map the PRADA elements to their curriculum, they need to learn to recognize PRADA elements within their discipline and see examples of how PRADA elements can be introduced. We believe that our approach to helping teachers learn about CT and how to integrate it into their classrooms provided them with opportunities to discover and elaborate on PRADA elements and to design their own learning segments with the assistance of computing experts. Teachers could enforce a long-term conversation about CT by identifying places in their existing curriculum where students are struggling and looking for opportunities to integrate CT and coding to help students learning.

Furthermore, we hypothesize that the PRADA model could also help teachers realize how the CT mindset can be used across disciplines to develop generalized problem-solving skills and dispositions. Research shows that interdisciplinary lessons give students the opportunity to apply knowledge from different areas in problem solving [14]. As our findings demonstrate, interdisciplinary teacher teams can mix and match PRADA elements from different content areas, which mitigates the concern that some PRADA elements might be hard to identify or explain in certain content areas. In addition, PRADA can help generalize CT across disciplines, providing students with a more tangible model with which to make sense of the problem-solving process and other CT approaches. While we have shown that teachers are successful at developing interdisciplinary lessons through PRADA, research into their pedagogical effects in classroom environments is still needed. Though our research team has developed and helped teachers implement CT lesson using the PRADA model[6], the broader potential impact of PRADA can only be fully realized when teachers can understand CT and implement effective CT-integrated lesson plans themselves.

In conclusion, this paper proposed and demonstrated how PRADA can be used to help K-12 teachers develop an understanding of Computational Thinking and how to integrate it into their disciplinary courses. We argue that PRADA is a CT model that can be easily understood by teachers, even those with little to no experience with programming or computer science. While the elements of CT are not novel, the way in which they were integrated into the teacher PD and the way teachers were asked to integrate them into their classrooms is novel. In the future, we will further investigate the impact of this promising PRADA model for supporting teachers in integrating CT into K12 classrooms.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Alfred V Aho. 2011. Ubiquity symposium: Computation and computational thinking. *Ubiquity* 2011, January (2011), 1.

[2] 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).

[3] Deborah Loewenberg Ball. 2000. Bridging practices: Intertwining content and pedagogy in teaching and learning to teach. *Journal of teacher education* 51, 3 (2000), 241–247.

[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 (Feb. 2011), 48–54. https://doi.org/10.1145/1929887.1929905

[5] Matt Bower, Leigh N Wood, Jennifer WM Lai, Cathie Howe, Raymond Lister, Raina Mason, Kate Highfield, and Jennifer Veal. 2017. Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education* 42, 3 (2017), 4.

[6] Veronica Cateté, Nicholas Lytle, Yihuan Dong, Danielle Boulden, Bita Akram, Jennifer Houchins, Tiffany Barnes, Eric Wiebe, James Lester, Bradford Mott, and Kristy Boyer. 2018. Infusing Computational Thinking into Middle Grade Science Classrooms: Lessons Learned. In *Proceedings of the 13th Workshop on Primary and Secondary Computing Education (WiPSCE '18)*. ACM, New York, NY, USA. https://doi.org/10.1145/3265757.3265778

[7] National Research Council et al. 2010. *Report of a workshop on the scope and nature of computational thinking*. National Academies Press.

[8] Barbara A Crawford. 2007. Learning to teach science as inquiry in the rough and tumble of practice. *Journal of research in science teaching* 44, 4 (2007), 613–642.

[9] International Society for Technology in Education (ISTE), Computer Science Teachers Association CSTA), et al. 2011. Operational definition of computational thinking for K-12 Education. online.

[10] Google. 2018. What is Computational Thinking? https://computationalthinkingcourse.withgoogle.com/unit?lesson=8&unit=1

[11] Cynthia L Greenleaf, Cindy Litman, Thomas L Hanson, Rachel Rosen, Christy K Boscardin, Joan Herman, Steven A Schneider, Sarah Madden, and Barbara Jones. 2011. Integrating literacy and science in biology: Teaching and learning impacts of reading apprenticeship professional development. *American Educational Research Journal* 48, 3 (2011), 647–717.

[12] 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.

[13] Maya Israel, Jamie N Pearson, Tanya Tapia, Quentin M Wherfel, and George Reese. 2015. Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education* 82 (2015), 263–279.

[14] Leslie Jarmon, Tomoko Traphagan, Michael Mayrath, and Avani Trivedi. 2009. Virtual world teaching, experiential learning, and assessment: An interdisciplinary communication course in Second Life. *Computers & Education* 53, 1 (2009), 169–182.

[15] Robin Jocius, Yihuan Dong, Deepti Joshi, Richard Robinson, Tiffany Barnes, Veronica Catete, Jennifer Albert, Ashley Andrews, and Nick Lytle. under review. Code, Connect, Create: The 3C Professional Development Model to Support Computational Thinking Infusion. *Computer Science Education* (under review).

[16] K12cs.org. 2018. Computational Thinking. https://k12cs.org/computational-thinking/

[17] Jane Margolis. 2010. *Stuck in the shallow end: Education, race, and computing*. MIT Press.

[18] Sharan B Merriam. 1988. *Case study research in education: A qualitative approach*. Jossey-Bass.

[19] Phil Sands, Aman Yadav, and Jon Good. 2018. Computational Thinking in K-12: In-service Teacher Perceptions of Computational Thinking. In *Computational Thinking in the STEM Disciplines*. Springer, 151–164.

[20] Anselm Strauss and Juliet M Corbin. 1990. *Basics of qualitative research: Grounded theory procedures and techniques*. Sage Publications, Inc.

[21] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2014. Defining computational thinking for science, technology, engineering, and math. In *American Educational Research Association Annual Meeting, Philadelphia, Pennsylvania*.

[22] 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.

[23] Stephen Wilson. 1977. The use of ethnographic techniques in educational research. *Review of educational research* 47, 2 (1977), 245–265.

[24] J Wing. 2011. Research notebook: Computational thinkingâĂŤWhat and why? The Link Magazine, Spring.

[25] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

[26] Aman Yadav, Sarah Gretter, Susanne Hambrusch, and Phil Sands. 2016. Expanding computer science education in schools: understanding teacher experiences and challenges. *Computer Science Education* 26, 4 (2016), 235–254.

[27] 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.