Professional Development for Computer Science Principles: Design Considerations and Teacher Learning Outcomes

Abstract

With the increased attention on integrating computer science concepts into K-12 curricula, there has been a growing investment into professional development opportunities that prepare teachers to teach computer science principles. Yet, little research exists on design features of professional development that help teachers gain the computer science content, skills and teaching pedagogy that ultimately make an impact on student learning and participation in the classroom. In this work we present a professional development model for helping K-12 teachers integrate computer science principles across the curriculum in a variety of content areas. We subsequently investigate the ways in which the design features of the model promoted teacher learning of computer science content and pedagogy.

Introduction

In recent years, there has been rekindled interest in K-12 computer science education, both in the U.S. and around the world. One aspect of computer science education that has gained momentum is computational thinking. Computational thinking combines critical thinking skills with the power of computing to help learners make decisions or solve problems. Wing (2006) argued that computational thinking is a fundamental skill for everybody and that "to reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" (p.33). This focus on helping all students develop computational thinking skills is also reflected in the newly released National Educational Technology Standards for Students (ISTE, 2016).

Fluck et al. (2016) identify three primary reasons for the rekindled interest in computer science. The first is the economic reason, which rests on the need for a country to produce a greater number of computer scientists. By 2018 it is projected that 51% of all STEM jobs will be in computer science related fields (Carnevale, Smith, & Melton, 2011). The second reason is social and emphasizes the value of helping all students become creators of computing rather than passive consumers. Despite the wide available of technology in teenage life, for instance, recent data indicate that most teens are only *users* of technology while only a small and fairly homogeneous number of teens acquire skills required to become *creators* of computing innovations (Cuny, 2012). The third is a cultural rationale, which enables people to be drivers of culture a change, by using computers to produce their own cultural artifacts. But there is also a fourth reason resting on the equity rationale. Specifically, there is urgency to broaden participation in computing particularly among females and under-represented minorities (Cuny, 2012). To address this issue, a new bold initiative called *CS for All* aims to empower all American students to learn computer science and be equipped with the computational thinking skills necessary for economic opportunity and social mobility.

Incorporating CS as a core element of the school curriculum, involves a number of challenges including what computer science content to teach and how to prepare teachers to acquire the knowledge and skills needed to teach that content (Angeli et al., 2016). In terms of content, a number of computer science curricula emerged in the last

few years as well as a framework focusing on Computer Sciences Principles (CSP) or big ideas in computing. These principles include: creativity, abstraction, data, algorithms, programming, Internet, and impacts of computing on society. Teachers, however, play a key role in how curricula are presented to students. Unfortunately, many teachers lack a strong foundation of computing and as a result effective professional development (PD) is key to helping teachers build capacity to use curricula and frameworks.

Purpose of the Study

With the increasing focus on integrating CS concepts into K-12 curricula, there has been a growing investment into PD opportunities that prepare teachers to teach computer science principles. Yet, little research exists on design features of PD that appear to help teachers gain the CS content, skills and teaching pedagogy that ultimately make an impact on student learning and participation in the classroom. In this work we present a PD model for helping K-12 teachers integrate CS principles across the curriculum in a variety of content areas. Specifically, we investigate the following questions:

- What design features should characterize computer science related PD?
- How do these design features impact teacher learning of computer science content and pedagogy?

Context of this Work

This work is situated in the context of a larger NSF-funded initiative which focuses on preparing middle and high school teachers interested in implementing CS principles in their classrooms, particularly in conjunction with STEM curricula. We describe the PD model developed as part of this work focusing on learning objectives, content, pedagogical strategies and job-embedded follow-up support. We subsequently report on teachers' input on their learning and remaining questions.

Description of the PD Model

Our approach to PD includes two key components: a 1-week long summer institute aimed at improving teacher computer science related knowledge and skills, and job-embedded follow up support during the academic year. Our 1-week summer institute, which is the focus of this work, is designed around core elements of effective PD reported in the literature, including: CS content, pedagogical strategies for teaching CS, and strategies for broadening participation in computing.

The focus of the summer institute is on materials aligned with CSP, specifically the seven principles of computing. Each day of the summer institute focuses on addressing content related to a particular principle through hands-on activities. The activities are delivered by content and pedagogy experts as well as teacher leaders who have previously engaged in the PD program. From a pedagogical standpoint participants learn about pedagogical strategies specific to delivering computer science lessons to students including pair programming, POGIL, open-ended projects allowing for creativity, team-based projects, a variety of kinesthetic activities such as those found on CS unplugged, pacing issues, and sustained reflection. On the third and fourth day of the institute teachers work in pairs or teams to draft lesson plans that integrate CS principles into existing school curricula which helps them draw connections between computer

science content, Common Core State Standards and the Next Generation Science Standards. All activities are conducted in a supportive and collaborative environment, which promotes active learning by engaging teachers in experiencing CS principles as learners. Figure 1 provides a snapshot of the week-long institute.

Time	Monday	Tuesday	Wednesday	Thursday
9:00-9:30		Pair Programming & Scratch Projects	Tales from the Field & CSTA discussion	Broadening Participation
9:30-10:15		Scratch Project Development	CSP: Data - Representation, Structures, & Manipulation (Binary Representations)	CIS: Internet (Routing Activity)
10:15-10:30	10-11: Registration and Check In	Break	Break	Break
10:30-12:00	11:00-11:30 Introductions, Program Purpose & Overview 11:30-12:00 CS Unplugged (Harold the Robot)	Lesson Plan Review & Mapping to Core Standards	CSP: Data - Lesson Activities (Finding Patterns in English, Working with sensor data)	CSP: Internet - web programming
12:00-12:45	Lunch	Lunch	Lunch	Lunch
12:45-1:15	CSP: Algorithms - Intro and Model Lesson (need to shrink from 45 to 30min)	CSP: Algorithms - CS Unplugged Activity (Battleship)	CSP: Data - CS Unplugged Activity (Databases)	CSP: Impact -Blown to Bits Presentations
1:15-2:15	CSP: Programming -Directed Scratch Lesson with Levels	Creating Recruiting devices: Interactive Projects with Scratch	CSP: Generating Questions from visualizations	Lesson Plan Review & Mapping to Core/ISTE/CSTA Standards
2:15-2:30	Break	Break	Break	Break
2:30-3:30	More Directed Scratch Lesson with Levels	CSP: Creativity - Assessing Programs for Creativity	CSP: Abstraction, Data & Models (Social Models)	Sharing of Implementation Plans
3:30-4:00	Reflection: Stop, Start, Continue; Reflection on Learning	Reflection: Stop, Start, Continue; Reflection on Learning	Reflection on Learning	Evaluation surveys; Resources; Going Forward
4:00	Adjourn	Adjourn	Adjourn	Adjourn
Homework	Blown to Bits Reading (1 assigned chapter)	How will integrate into classroom	How will integrate into classroom	

Figure 1. Snapshot of Week-Long Summer Institute

Participants

Participants included 22 middle/high school teachers from over a dozen different schools. These teachers taught a variety of content areas including mathematics, science, technology, business education, engineering or other STEM related field.

Data Collection and Analysis

Data for all teachers were collected from two sources: pre and post self-assessed surveys on their comfort level with CS principles, and daily reflections on PD activities from the summer institute. All reflections were structured around the same three prompts to facilitate consistent responses: (a) What did you learn from this session? (b) What do you still have questions about? and (c) What additional supports could you use? This allowed us to document the ways in which participation in the summer institute can support changes in teacher learning and instructional practice.

Participants' surveys were analyzed using descriptive statistics. Daily reflections from the summer institute (N = 88) were analyzed qualitatively using the constant comparative method (Bogdan & Biklen, 2003). Specifically, we repeatedly read reflections in order to identify similarities and differences among participant responses as well as emergent themes specific to computer science content and pedagogy presented in the summer institute (Miles & Huberman, 1994).

Findings

In this section, we present the findings of our work organized in four areas: (a) teacher learning of computer science content, (b) teacher learning of pedagogical strategies for teaching computer science content, and (c) teacher learning of strategies for broadening participation in computing.

Pre and post survey data (see Table 1) as well as analysis of daily reflections indicated that teachers became more confident in teaching computer science skills related to CS principles (creativity, abstraction, data, algorithms, programming, Internet and impacts). In their daily reflections, most teachers also indicated learning new content related to algorithms as well as the difference between algorithms and programming. Further, they indicated that they had improved their computer programming skills in Scratch and HTML. As one teacher noted, *"I have never coded before. Today I learned to do that with HTML and it was great!"* Finally, most teachers indicated improvements in their knowledge and skills around data analysis and visualization techniques.

Table 1. Pre/Post Survey Data on Confidence in Teaching CS Principles

I feel more confident teaching computer science skills related to …	Pre: Mean	Pre: SD	Post: Mean	Post: SD
Creativity	3.10	1.32	4.50	0.74
Abstraction	2.83	1.26	4.43	0.79
Data	3.10	1.37	4.19	1.10
Algorithms	3.10	1.21	4.38	0.84
Programming	3.10	1.35	4.43	0.73
Internet	3.55	1.25	4.48	0.73
Impacts	3.41	1.43	4.38	0.72

N=22

1: strongly disagree to 5: strongly agree

Despite improvements in their knowledge and skills, participants also identified remaining questions particularly around abstraction and technical skills in using Scratch programming. They also indicated the need for more tutorials on how to analyze data and more content specific ideas for incorporating data into their curricula.

From a pedagogical standpoint, participants really valued the opportunity to experience CS unplugged activities as learners (e.g., Battleships: http://csunplugged.org/searching-algorithms/; Harold the Robot: http://csunplugged.org/harold-the-robot-2/). They also indicated that they learned new pedagogical strategies for teaching CS, including pair programming and inquiry oriented activities. Further, all participants appreciated the engagement with existing content standards in mathematics, literacy and science and the ways in which they were able to draw connections between CS principles across disciplines. A number of teachers discussed learning about the importance of using rubrics to assess computational products, an activity that they practiced during the week-long institute. Finally, several teachers indicated improvements in their ability to generate enthusiasm among students for CS, including girls and minorities.

When asked which activities they planned to implement in their classrooms, participants uniformly noted pedagogical activities such as CS unplugged and pair

programming. They also indicated ways in which they plan to incorporate data (e.g., by utilizing Google forms), and programming with Scratch in their curriculum. Some teachers also indicated that they plan to have teams of students debug existing programs. As one teacher noted, *"This the first PD I had where I am not overwhelmed and actually have materials I can directly use in the classroom."*

Conclusion

In this work we presented one approach to PD that helps teachers integrate CS principles into existing curricula. We argue that such integration is essential for broadening participation in computing and promoting computational thinking skills across the entire education pipeline. Our findings indicate that the proposed PD model holds promise for the successful infusion of CS content in middle and high school classrooms as it helps teachers gain confidence in their own knowledge (content and pedagogical) as well as develop plans for implementing CS principles into existing curricula. Our next challenge is scaling up the model to reach a greater number of teachers in our region.

References

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. *Educational Technology & Society, 19* (3), 47–57.

Bogdan, R. C., & Biklen, S. K. (2003). *Qualitative research for education: An introduction to theories and methods* (4th ed.). New York, NY: Pearson.

Carnevale, A.P., Smith, N. & Melton, M. (2011). *STEM: Science Technology Engineering and Mathematics*. Georgetown University: Center for Education and the Workforce.

Cuny, J. (2012). Transforming high school computing: A call to action. *ACM Inroads*, 3(2): 32-36.

Fluck, A., Webb, M., Cox, M., Angeli, C., Malyn-Smith, J., Voogt, J., & Zagami, J. (2016). Arguing for Computer Science in the School Curriculum. *Educational Technology & Society, 19* (3), 38–46.

ISTE (2016). The 2016 ISTE Standards for students. Retrieved from http://www.iste.org/standards/standards/for-students-2016

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook*. Thousand Oaks, CA: Sage.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.