Integrating computational thinking in elementary classrooms: Introducing a toolkit to support teachers.

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

Rachel Larimor
College of Education, Michigan State University, East Lansing, MI, USA
larimor5@msu.edu

Kathryn M. Rich
College of Education, Michigan State University, East Lansing, MI, USA
richkat3@msu.edu

Christina Schwarz
College of Education, Michigan State University, East Lansing, MI, USA
cschwarz@msu.edu

Cite as: Yadav, A., Larimore, R., Rich, K., Schwarz, C. (2019). Integrating computational thinking in elementary classrooms: Introducing a toolkit to support teachers. In *Proceedings of Society for Information Technology & Teacher Education International Conference* 2019. Chesapeake, VA: AACE.

Abstract: Driven by the need for students to be prepared for a world driven by computation, a number of recent educational reforms in science and mathematics have called for computational thinking concepts to be integrated into these content areas. However, in order for computational thinking (CT) to permeate K-12 education, we need to educate teachers about what CT ideas are and how they relate to what happens in their classroom on a day-to-day basis. This paper presents a toolkit to scaffold elementary teachers' understanding of computational thinking ideas and how to integrate them into their lesson plans.

Keywords: Computational thinking; K-12

Introduction

Recent educational reforms in K-12 education, such as the Next Generation Science Standards (NGSS) and Common Core State Standards (CCSS), either explicitly or implicitly call out the need for students to engage in computational thinking. Computational thinking (CT) has been defined as "breaking down complex problems into more familiar/manageable sub-problems (problem decomposition), using a sequence of steps (algorithms) to solve problems, reviewing how the solution transfers to similar problems (abstraction), and finally determining if a computer can help more efficiently solve those problems (automation)" (Yadav, Hong, Stephenson, 2016).

This push to bring CT to K-12 has been driven by the fact that the world students are growing up in and will be working in is driven more and more by computation. A recent report argued that with computing technologies transforming many dimensions of our personal and work life, computational thinking is a critical part of "what is important to know and know how to do it in a computational world" (Digitial Promise, 2017 p. 4). This has led a number of organizations, such as International Society for Technology in Education (ISTE) and Computer

Science Teachers Association (CSTA), to develop standards for educators and students in computational thinking. And as Denning (2017) suggested, these standards do not consistently define CT and associated concepts in similar ways. As educators participate in different professional development workshops, it could lead to confusion on what CT is and how to bring CT ideas to their classrooms.

Research on how in-service teachers conceptualize computational thinking has suggested that they often believe that computational thinking involves doing mathematics, logical thinking, problem solving, using computers, coding, or simply using technology in the classroom (Sands, Yadav, & Good, 2018). While some of these ideas overlap with computational thinking, teachers still lack an understanding of how to bring these ideas to their classrooms. For example, referencing computational thinking as coding is accurate, but it limits the power of CT ideas and how they could be applied to other disciplines. While computational thinking is central to computer science (Denning, 2017; Wing, 2006), we also see it an an important skill for students to learn within the context of other disciplines (Yadav, Hong, Stephenson, 2016). We need to start engaging kids in these ideas and practices starting at the elementary level.

In order for computational thinking to permeate K-12 education, we need to educate teachers in what these ideas are and how they relate to what happens in their classroom on a day-to-day basis. While there is work starting to emerge on teacher training at the high school level, there has been less attention paid to how to bring CT to elementary classrooms. Furthermore, elementary schools face additional pressures of accountability as a result of No Child Left Behind; thus, more resources are being devoted to standardized test subject areas of language arts and mathematics (Marx & Harris, 2006). This means there is limited time during the school year to add new initiatives like computer science even as other core subjects like science are being pushed out (Marx & Harris).

One approach to bring CT to elementary classrooms has been to work within the constraints of K-12 systems and integrate it within core subject areas, such as mathematics and language arts. We believe that unplugged approaches - without the use of computers or other technology - provide an easy on ramp for elementary teachers to embed computational thinking in the curriculum. As teachers get more comfortable and see CT connections within their lessons, we can scaffold teacher learning on integrating plugged CT activities using computational tools and environments. Starting with unplugged activities lessens the cognitive load that comes with learning computational tool as well as understanding how CT ideas connect to core subject areas.

The focus on CT ideas first using unplugged approaches has been shown to to develop elementary teachers' understanding of how these ideas connect and fit within their classrooms (Yadav, Krist, Good, & Caeli, 2018). Specifically, Yadav and colleagues examined how unplugged approaches to CT could provide elementary teachers with opportunities to embed CT within science. Using teaching vignettes, the authors examined how teachers' understanding of CT shifted over the course of a year. Results suggested that teachers began with ideas of CT as generalized and broad, but their views of CT shifted to be more sophisticated and elaborate versions of those ideas.

While some work is beginning to emerge on how to engage K-12 teachers in computational thinking ideas, there is much debate on how to do so. For example, Denning (2017) argued that engaging in computational modeling is at the heart of computational thinking and that we engage in CT ideas, such as abstraction and decomposition, to get a model to accomplish a certain work. Similarly, a recent blogpost summarized Twitter discussion on whether computational thinking exists and how it could be brought to K-12 classrooms (Guzdial, 2018). The debate centered around whether we can truly expose students to computational thinking ideas without coding, which is how computer science is primarily introduced in primary and secondary schools.

We don't disagree with these views on the need for computational models and coding as vehicles to introduce CT. We do not, however. see coding as the best starting place in elementary schools. As noted above, we think this approach places a significant cognitive load on teachers that could be alleviated by introducing CT ideas through unplugged contexts.

What do these CT ideas mean, and how can elementary teachers bring them into their classrooms? In this paper, we will describe four computational thinking ideas (abstraction, decomposition, patterns, and debugging) that we have productively used with elementary teachers to bring CT into their mathematics and science lessons as a part of a National Science Foundation funded project. Other CT concepts, such as algorithms, are present within these four ideas and not explicitly called out given the context of our work - math and science in elementary schools in linguistically and racially diverse urban settings.

A Teacher Toolkit: Scaffolding CT integration

We developed a computational thinking teacher toolkit to scaffold integration of the four CT concepts (abstraction, decomposition, patterns, and debugging) in elementary classrooms. The toolkit included description and essential features of each CT concept as well as list of questions teachers could use to engage students in the computational ideas. In addition, we also developed a lesson screener tool that teachers could use to identify where CT concepts were already present in their lessons and ways they could enhance their lessons by more bringing CT concepts more explicitly. Below is a summary of the four CT concepts we focused on and how teachers could bring each idea into their classrooms.

Abstraction
Abstraction is about reducing complexity or identifying general principles that can be applied across situations or problems.

1. Encourage students to focus on the most important information and hide unnecessary detail.
2. Provide opportunities for students to represent problems/phenomena in ways that simplify them.
3. Encourage students to identify principles that can be applied across situations/problems.

Decomposition
Decomposition is about managing complex tasks or situations by breaking them down into smaller, more manageable parts. Students can use decomposition to approach problems that, at first, may seem intimidating.

1. Provide opportunities for students to break down a phenomenon or object into parts.
2. Choose tasks where students can break down the problem in multiple ways.

Patterns
Patterns are everywhere. We see them every day. You can engage students in patterning by having them recognize and form patterns.

1. Ask students to look for and discuss patterns during activities.
2. Provide opportunities for students to generate and describe patterns.

Debugging
Debugging is about finding and fixing errors. Sometimes it is called troubleshooting.

1. Encourage students to "debug" when something doesn't work as they had expected or planned.
2. Avoid the urge to fix problems for students. Allow them to reason through courses of action for themselves.

Scholarly significance

With a number of current educational frameworks, such as the Next Generation Science Standards (NGSS) and Common Core State Standards (CCSS) highlighting the need for students to be exposed to computational thinking in the K-12 curriculum, this toolkit provides a mechanism for researchers to work with teachers to integrate CT in their classrooms. For example, teachers could bring the CT concept of abstraction into their science instruction when students develop and use models (data models as well as theoretical models), by having students represent phenomena in ways that simplifies it. We see this as a starting place. Future work should examine how teachers use the toolkit to integrate CT within their lessons.

Acknowledgement: This work is supported by the National Science Foundation under grant number 1738677. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33-39. DOI: 10.1145/2998438

Digital Promise (2017). Computational thinking for a computational world digital. Retrieved from https://digitalpromise.org/initiative/computational-thinking/

Marx, R., & Harris, C. (2006). No child left behind and science education: Opportunities, challenges, and risks. *The Elementary School Journal, 106*(5), 467-478. DOI: 10.1086/505441

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

Sands, P., Yadav, A., & Good, J. (2018). Computational thinking in K-12: In-service teacher perceptions of computational thinking. In M. S Khine. (Ed.). *Computational Thinking in the STEM Disciplines* (pp. 151-164). Springer.

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

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding a 21st century problem solving in K-12 classrooms. *TechTrends 60*, 565-568. DOI: 10.1007/s11528-016-0087-7.