TechTrends (2016) 60:565–568
DOI 10.1007/s11528-016-0087-7

ORIGINAL PAPER

Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms

Aman Yadav[1] · **Hai Hong**[2] · **Chris Stephenson**[2]

Published online: 30 May 2016

Abstract The recent focus on computational thinking as a key 21st century skill for all students has led to a number of curriculum initiatives to embed it in K-12 classrooms. In this paper, we discuss the key computational thinking constructs, including algorithms, abstraction, and automation. We further discuss how these ideas are related to current educational reforms, such as Common Core and Next Generation Science Standards and provide specific means that would allow teachers to embed these ideas in their K-12 classrooms, including recommendations for instructional technologists and professional development experts for infusing computational thinking into other subjects. In conclusion, we suggest that computational thinking ideas outlined in this paper are key to moving students from merely being technology-literate to using computational tools to solve problems.

Keywords Computational thinking · K-12 · Computer science education · Teachers

Introduction

Computer science plays a vital role in today's technology and globally connected world, which means that we need to introduce computing ideas to students early during their schooling years. Google (2015) found that while students, parents, teachers, and administrators value computer science education, school administrators significantly underestimate parental demand for access to computer science learning for all students and instead focus attention and resources on subject areas that require mandatory testing. Given the recent focus of educational reform movement on standardized testing of core subject areas (such as, mathematics, reading) and that computer science is not a required subject area, this is not a surprising finding. Furthermore, administrators have also reported that computer science is not always prioritized by their school boards and the lack of qualified teachers and resources to train or hire teachers makes it difficult to offer computer science (CSTA, 2013; Google, 2015). The challenges of offering computer science courses are even more stark for rural and small school districts where administrators are less likely to agree that their school board thinks computer science is important, and less likely to say it is a top priority. The question then is how do we address the need to introduce K-12 students to computer science ideas while operating within the constraints of available resources and focus on standardized testing? How do we enable teachers in the core subject areas to embed computing in the curriculum?

Computational thinking (CT) offers an encompassing approach that exposes students to computing ideas and principles in the context of the subject areas they are already learning. Wing popularized the term in her 2006 Communications for the ACM article arguing, "To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" (Wing 2006, p. 33). What is computational thinking? The essence of computational thinking involves 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 us

✉ Aman Yadav
ayadav@msu.edu

[1] Michigan State University, 620 Farm Lane, East Lansing, MI 48824, USA

[2] Google Inc, 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA

more efficiently solve those problems (automation). These computational thinking steps are foundational to computer science, but their power and utility extend far beyond any single discipline. Mishra and Yadav (2013), for example, argued that "computational thinking can foster creativity by allowing students to not only be consumers of technology, but also build tools that can have significant impact on society" (p. 11). Furthermore, computational thinking aligns with the need for students to become media information literate, which includes understanding how information and data can be represented to convey different meaning (Wilson et al., 2013).

The following sections elaborate on computational thinking constructs and how they relate to K-12 learning, and present resources and examples that can help educators easily and effectively embed CT in their classrooms.

Computational Thinking and Schooling

Wing (2006) argued that computational thinking involves three key constructs: Algorithms, Abstraction, and Automation - the three A's of CT. An *algorithm* (much like a recipe) is a step-by-step series of instructions. *Abstraction* involves generalizing and transferring the problem solving process to similar problems (Barr and Stephenson 2011). Finally, *automation* involves using digital and simulation tools to mechanize problem solutions. In their effort to make computational thinking more applicable to K-12, the Computer Science Teachers Association (CSTA) and International Society for Technology in Education (ISTE) developed an operational definition of computational thinking that includes nine core CT concepts and capabilities (Barr and Stephenson 2011). These core computational thinking ideas include: data collection, data analysis, data representation, problem decomposition, abstraction, algorithms & procedures, automation, parallelization, and simulation.

There have been a number of efforts to embed computational thinking in K-12 classrooms based on these computational thinking competencies. The majority of these efforts have focused primarily on computer science curricula such as the proposed Advanced Placement Computer Science Principles (CSP) course, the Exploring Computer Science course, and programming tool use (such as, Code.org's block-based programming environment). The CSP course is the largest of these efforts with the new course set to launch in Fall 2016. The CSP framework proposes six computational thinking practices to "help students coordinate and make sense of knowledge to accomplish a goal or task" (College Board 2014, p. 2). The six practices are designed to allow students to: better understand the influence of computing and its implications on individuals and society (*connecting computing*); engage in computing by designing and developing computational artifacts (*creating computational artifacts*); apply abstraction to develop models and simulations of natural and artificial phenomena (*abstracting*); develop solutions, models, and artifacts for problems (*analyzing problems and artifacts*); describe the influence of technology and computation supported by data visualizations (*communicating*); and learn to work together effectively to solve ill-structured problems using computation (*collaborating*). In sum, the idea of CT is to allow students to develop a foundational understanding of computing and develop competencies that move them from being users of technology to producers of information technology (Yadav et al. 2014).

While it is valuable for students to learn computational thinking within the context of computer science curricula and programming environments, the constraints of a K-12 classroom might not make it feasible for all schools to have access to standalone computing courses. However, computational thinking ideas are cross-disciplinary and can be embedded across the elementary and secondary subject areas. Furthermore, current educational reforms, such as Next Generation Science Standards (NGSS) and Common Core also highlight the need for students to be exposed to computational thinking in the K-12 curriculum. The nine core concepts and capabilities from CSTA/ISTE framework provide a good place to begin to embed CT in the K-12 core content areas. For example, one of the key components of computational thinking is using an *algorithm*, which involves using a sequence of steps in an orderly manner to solve problems or complete tasks. We use algorithms everyday in our lives from following a cooking recipe to giving directions from point A to point B. Algorithms are a key skill in order to develop students' "ability to interpret the world as algorithmically controlled conversions of inputs to outputs" (Denning 2009, p. 29). For example, elementary students can learn about algorithms by breaking down a simple daily task, such as brushing teeth, into a sequence of steps (Yadav et al. 2014). Similarly, students in second language classes could use cooking recipes to learn about algorithms (Deschryver and Yadav 2015).

Another set of CT concepts that provides an opportunity to introduce computational thinking concepts across core content areas is data collection/data analysis/data representation. For example, a social studies teacher can use data from most-used words from presidential inaugural speeches from 1789 to 2009 (Source: http://nyti.ms/1XLe26x) and have students analyze differences between the speeches across an era or between Democratic and Republican presidents. Similarly, science teachers can have students explore international and United States greenhouse emission data sets from Google Public Data Explorer (http://www.google.com/publicdata) to compare emission rates across states, countries, and even economic sectors (agriculture, energy, industrial processes, and waste). The ability to make sense of data to identify

solutions to problems is a key computational thinking skill. It can also provide access to significant career opportunities given that jobs requiring computing skills are expected to grow from 1.9 million to 4.4 million by 2017 alone (Big Data Jobs Index, 2016).

Abstraction is another key computational thinking idea that can be embedded in the classroom. Abstraction involves the ability to generalize and transfer a solution from one problem to other similar problems. Abstraction also involves developing and representing models of the real world (such as using a physical model that represents our solar system or a simulation that shows how populations respond to situations such as disease outbreaks). Building on our prior example of data analysis and representation, abstraction could be practiced in science classrooms by teachers engaging students in analyzing data to draw conclusions and develop general principles.

While the CT concepts and practices discussed above can be implemented in a regular classroom, one of the key CT components is *automation*, which requires access to computing tools. One way to engage students in automation is through modeling and simulation. A number of tools and curricula are available to teachers who want to address computational thinking concepts through simulation. NetLogo (https://ccl.northwestern.edu/netlogo/), for example, provides a plethora of simulations and models that could be embedded in content areas such as earth science, biology, chemistry, social studies, physics, and social science to help students see how patterns emerge. For example, a social studies teacher could use the model to examine how individual preferences to live in proximity to populations similar to themselves can have a ripple effect leading to a large-scale segregation pattern. Barr and Stephenson (2011) provide specific examples of how the CT concepts and capabilities from the CSTA/ISTE framework could be implemented in specific subject areas in K-12, such as language arts, social studies, science, mathematics, and computer science.

It would be unreasonable to expect teachers to incorporate computational thinking concepts into their practice without support and opportunities to apply these ideas to authentic tasks (Yadav et al. 2013, 2014). We believe that these support mechanisms need to be continuous rather than one- or two-time professional development events. One way to provide these professional development opportunities is via online learning reinforced with communities of practice. Google, for example, provides a free online course called Computational Thinking for Educators (https://computationalthinkingcourse.withgoogle.com/course). This course is structured around algorithms and patterns for four groups of teachers - humanities, mathematics, science, and computer science teachers. In addition to online materials, it offers a community of practice that provides a place for educators to share ideas, challenges, solutions, and resources learned from their classroom practice. Michigan State University also offers a computational thinking course for educators as a part of its Masters in Educational Technology (MAET) program. The course is designed to enable teachers to explore ideas that support the development of their CT skills and pedagogical approaches to incorporating CT in their own classrooms. The teachers work collaboratively on a semester-long project to develop meaningful activities relevant to their own contexts (such as, subject area, grade level, and school resources) that allow them to integrate computational thinking at their schools.

Conclusion

Recent national efforts have emphasized the importance of computational thinking to prepare students to succeed and thrive in our increasingly technological society, to maintain U.S. economic competitiveness, to support inquiry in other disciplines, and to enable personal empowerment to tackle complex problems (National Research Council 2011). Given that computational thinking focuses on problem solving and engages students in designing processes that can be automated, it is essential that K-12 educators and administrators explore ways to embed CT ideas into their curricula and practice. Given the challenges of meeting today's curricular demands, we believe that connecting computational thinking ideas to what teachers already do in their classrooms is the best approach. The CSTA/ISTE framework offers one approach to integrating computational thinking constructs and capabilities within the context of the existing content areas. While research in this area is relatively new, there are promising results that highlight the positive impact of CT ideas on student outcomes in traditional K-12 subject areas. For example, a recent study found that when computational thinking ideas were embedded in a sixth grade mathematics classroom, students' understanding of mathematical processes increased significantly when compared to students in the control group (Calao et al. 2015). These findings highlight that computational thinking can not only impact students' problem solving skills in general, but also have a significant influence on their academics (Calao, Moreno-Le´on, Correa, & Robles).

In order to achieve the goal of computational thinking for all, it is important to provide professional development opportunities that are tied to teachers' curricular needs in their subject areas. Specifically, computer science educators, teacher educators, and educational technology faculty need to collaborate with teachers to develop activities that make visible the inherent overlap of computational thinking idea and practices with subject area concepts. As discussed previously, the CSTA/ISTE framework provides a starting point for professional development experts to work closely with teachers and instructional technologists at the K-12 level. In summary, professional development for teachers must go hand-in-hand with

the curriculum and lesson development plans that work within the local context of the classroom.

While working with inservice teachers to embed CT in elementary and secondary classrooms is important, we also need to introduce computational thinking for preservice teachers in their teacher education programs. Within teacher education curricula, preservice teachers could learn about computational thinking in core courses (such as, learning theory and educational technology courses) and then expand their understanding about how computational thinking applies in a particular subject through teaching methods courses in their licensure areas.

The kinds of changes that we are advocating require vision and leadership from the instructional technology community and professional development experts. Members of this community can play a pivotal role by helping to articulate the connection between computational thinking and all academic disciplines, developing content to support integration into curricula, and taking the lead in designing and facilitating both preservice and inservice opportunities for learning. They can also help drive critical conversations with and between leaders in all academic subject areas.

In conclusion, we believe that the computational thinking ideas outlined in this paper are key to moving students from merely being technology-literate to using computational tools to solve problems and represent knowledge. Developing teachers' understanding of computational thinking and highlighting connections to their curricular context is key to successfully embedding CT in K-12 classrooms. Our work with preservice teachers has shown that they see the relevance of these ideas in the classroom and can learn to adopt them in their own curriculum (Yadav et al. 2014). This effort also requires buy-in from the K-12 administrators, including superintendents, principals, and school board members to provide necessary resources and tools for teachers. These resources might include, release time for professional development, access to computing tools in the classrooms, and engaging with computational thinking leaders to discuss approaches to CT in the classrooms. For teachers interested in learning more about these ideas, Google's Computational Thinking for Educators course serves as a good place to start and connect with a community of teachers with similar teaching interests.

References

Barr, V., & Stephenson, C. (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), 48–54.

Big Data Jobs Index (2016). Retrieved from https://icrunchdatanews.com/big-data-jobs-index/.

Calao, L. A., Moreno-Le´on, J., Correa, H. E., & Robles, G. (2015). Developing mathematical thinking with scratch an experiment with 6th grade students. In *Design for teaching and learning in a networked world* (pp. 17–27). Springer International Publishing.

College Board. (2014). AP computer science principles draft curriculum framework. Retrieved June 26, 2015, from https://advancesinap.collegeboard.org/stem/computer-science-principles.

CSTA (2013). B*ugs in the System: Computer science teacher certification in the U.S.* New York, NY: Computer Science Teachers Association.

Denning, P. J. (2009). The profession of IT beyond computational thinking. *Communications of the ACM, 52*(6), 28–30.

Deschryver, M. D., & Yadav, A. (2015). Creative and computational thinking in the context of new literacies: working with teachers to scaffold complex technology-mediated approaches to teaching and learning. *Journal of Technology and Teacher Education, 23*(3), 411–431.

Google (2015). *Searching for computer science: Access and barriers in U.S. K-12 education*. Retrieved from https://services.google.com/fh/files/misc/searching-for-computer-science_report.pdf.

Mishra, P., & Yadav, A. (2013). Of art and algorithm: rethinking technology & creativity in the 21st century. *TechTrends, 57*(3), 10–14. doi:10.1007/s11528-013-0668-7.

National Research Council. (2011). *Report of a workshop of pedagogical aspects of computational thinking*. Washington, DC: The National Academies Press. http://doi.org/978-0-309-21474-2

Wilson, C., Grizzle, A., Tuazon, R., Akyempong, K., & Cheung, C. K. (2013). *Mediaand information literacy curriculum for teachers*. UNESCO. Retrieved from http://www.unesco.org/new/fileadmin/MULTIMEDIA/HQ/CI/CI/pdf/media_and_information_literacy_curriculum_for_teachers_en.pdf.

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

Yadav, A., Hambrusch, S., Korb, T., & Gretter, S. (2013). Professional development for CS teachers: A framework and its implementatiod. In *Future directions in computing education summit*. Orlando, FL. Retrieved from http://web.stanford.edu/~coopers/2013Summit/attendees.html.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education, 14*(1), 1–16.