Computer science for all
Computer science and computational thinking for STEM integration

Paige **Prescott**
PhD student, Organization, Information and Learning Sciences, University of New Mexico
Albuquerque, New Mexico
pprescott@unm.edu

Abstract

There is increasing pressure on schools to expand access to computer science education. Governments are rapidly making policies to include computer science as part of their national curriculum and to require schools to address computational thinking as well. Integration of computational thinking concepts into existing content area courses such as math and science is seen as authentic to the computer science field where science and math professionals are increasingly using computation in their work. This paper is a literature review which starts with an exploration of the literature relating to computer science education, research in this field as well as computational thinking. It ends with a focus on the literature connected to integrating computer science and computational thinking into Science, Technology, Engineering and Math (STEM) classes and gives suggestions for areas in need of more advanced research.

Keywords: computational thinking, STEM, computer science education

Introduction

Computer science (CS) and computational thinking (CT) are increasingly becoming part of education K-12. In recent years, the expansion of computer science education resources has allowed schools to bring these topics into the classrooms to allow students at a young age to start learning these concepts. With the growth of the technology-related jobs, there is growing pressure to keep expanding the computer science options in schools. Most of the computer science experiences for students started in the informal education space through summer camps and afterschool clubs. This avenue helped to create foundational practices for both teachers and researchers. However, there is more need to understand how computer science and computational thinking can be part of the in-school experience for all students. Integrating computer science activities into content classes is a rich area for expansion and has the potential to impact all students in a school as opposed to the select few that participate in out-of-school

experiences. The newest science standards for the United States, Next Generation Science Standards (NGSS) has identified computational thinking as an important part of scientific inquiry and is one of the eight scientific practices that all students should be engaged in while learning science content (Lead States, 2013).

The goal of this paper is to explore the literature of CS and CT education research in order to understand the current state of this field of research as it pertains to integrated learning experiences for students in elementary or middle school STEM classes in which the computing content is not separate from the lesson but is intrinsic to acquisition of the core content knowledge such as science or engineering design in addition to CT and CS.

Computer science and constructionism

The concept of CS education as a foundation, belonging in the educational setting, began with Perlis' 1961 MIT Sloan School lecture "Computers and the World of the Future" where he insisted that all undergraduates should learn to program because he saw it as a foundational understanding needed as the automation of processes was certain to become more ubiquitous (as cited in Guzdial, 2008). Seymour Papert, fresh from Piaget's educational lab, embarked on a new era of education in which he combined the new concepts of computer programming through the lens of Piaget's constructivism to create a new paradigm which he called 'constructionism.' In the early 1970's he created a new computer programming language, Logo, to use as an educational tool. His influential work, "Mindstorms: Children, computers and powerful ideas" lead to a quiet revolution in education that continues to inform and inspire educational initiatives to bring computer science into schools. Despite its publication in 1980, the ideas Papert writes about still feel modern and fresh compared to where much of formal education is now, with siloed content areas and teacher-directed learning. He continued to refine constructionism, establishing his Media Lab at MIT in 1985 which emphasized play and learning as inextricably linked. Constructionism is similar to constructivism in that learning is done through building on existing knowledge structures (Papert & Harel, 1991). However, constructionism differs in that the "constructionism boils down to demanding that everything be understood by being constructed." (p. 2) and is informed by his experiences watching art classes and, in turn, how students were learning to program graphics by teaching themselves angles in which 'learning-by-making' is the simplest definition (Papert & Harel, 1991). This approach to education was formalized in a partnership with Lego in which programmable robotics kits used Logo and were based on the ideas from MIT's Media Lab. This connection between programming and physical devices lead to a large infiltration of robotics into schools as learning devices and can be seen in modern kits such as Lego's EV3, Vex, Dash & Dot, Ozobots, Spheros, mBot, and hundreds of other variations.

Papert's work still resonates and has informed generations of computer scientists and educational researchers not only in his philosophy of education but also in computer science educational software design. Kafai and Resnick (1996) continued to build on the concepts embedded in constructionism and came up with three key principles for designing computer-based learning environments in computer science: create a learning culture, design tools with powerful ideas and allow for personal expression. In 2007, the Lifelong Kindergarten group at the MIT Media Lab launched the Scratch programming environment with the intent that it be "tinkerable, more meaningful, and more social than other programming environments"(Resnick

et al., 2009). This approach led to an explosion of use in educational settings and created the foundational experience in computer science for many young students.

CT Defined

The work of Papert, DiSessa, Kafai and Resnick lead directly towards the concept of CT in which problem-solving techniques are ultimately put into practice through computers. Although Jeannette Jeannette M Wing (2006) is often attributed with coining the phrase 'computational thinking' it was, in fact, Seymour Papert who first wrote about it in 1996 (Papert, 1996). The concepts of CT have been embraced relatively rapidly into the education lexicon but there isn't a deep consensus on its definition nor how it is measured or evaluated programmatically.

CT was initially articulated as a skill that everyone, not just computer scientists, should have because it involves "problem solving, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science."(Jeannette M Wing, 2006) Wing also calls out the concepts of abstraction and decomposition as fundamental to computational thinking. However, as the concept matured, no clear agreement developed around the topic (Barr & Stephenson, 2011; National Research Council, 2010). Brennan and Resnick (2012) recognized the lack of consensus around this topic and developed a framework for CT based on their study of 'interactive media designers' on Scratch. There are three key dimensions for their framework: computational concepts, computational practices and computational perspectives. Computational concepts are further defined into seven topics, mostly relating to computer science concepts, which are: sequences, loops, parallelism, events, conditionals, operators and data. They then further define these concepts and give examples using the Scratch code blocks.

Researchers and thought leaders in CT have tended to come back to the core four concepts of abstraction, decomposition, automation and analysis (CSTA, 2011; Lee et al., 2011; National Research Council, 2010; J.M. Wing, 2011; Yaşar, 2017). Alongside these definitions many also include dispositions for student that include persistence, collaboration, dealing with open-ended problems and ambiguity (Barr & Stephenson, 2011; CSTA, 2011; Lee et al., 2011).

The concepts of CT are often confused with just learning to program a computer because of the emphasis on the coding environment that is used (Aho, 2012; Lee et al., 2011; Voogt, Fisser, Good, Mishra, & Yadav, 2015). However, other researchers have tried to decouple CT from programming and have focused on activities that emphasize non-computer or 'unplugged' lessons to understanding some of these concepts (Brackmann et al., 2017).

CT is flexible enough to be approached through integration into different content area such as science, math and media arts (Lead States, 2013; Orton et al.; Schanzer, Fisler, Krishnamurthi, & Felleisen, 2015; Wagh, Cook-Whitt, & Wilensky, 2017; Weintrop et al., 2016; Wilensky, Brady, & Horn, 2014). And can also be done in the informal education space through afterschool clubs and summer camps (Denner, Werner, & Ortiz, 2012; Gallup, 2015; Kafai, Peppler, & Chiu, 2007; Repenning & Ioannidou, 2008).

CT Integration in STEM

Computer science (CS) is often taught in a decontextualized manner in which the CS topics are presented in a CS class that is separate from other content areas. This approach is problematic in that students do not see the usefulness and connection of CS across many fields and do not perceive CS as a creative and applicable topic that can help solve problems in various

contexts. When CS is applied in science or mathematics classes there is greater understanding of the content concepts as well as CS concepts than when taught as separate topics (Blikstein, 2012; Guzdial, 1994; Orton et al.; Schanzer et al., 2015; Webb, Repenning, & Koh, 2012). Thus, integrating CS into existing classes is an important strategy when considering options for implementing CS in diverse educational settings.

CT education has been approached through informal learning as well as during the school day. The newest science standards from the United States are Next Generation Science Standards (NGSS) which explicitly calls out CT in their practices. There are many that are seeking to understand how CT can happen in a regular science classroom (Lead States, 2013). Some, like Project GUTS (Growing Up Thinking Scientifically), have designed curriculum that is aligned to both the NGSS and the Computer Science Teachers Association computer science standards that emphasizes computational modeling as a tool to engage students in CT (Lee et al., 2011). This approach is supported by Wilensky et al. (2014) who see that integrating CT practices within science classes with be a more authentic experience in terms of how CT is used by professional scientists. Other studies have shown that integrating CT and science not only advances the students' knowledge in CT but also promotes deeper learning of science concepts (DiSessa, 2001; Grover, Pea, & Stephen, 2015; Weintrop et al., 2016; Wilensky et al., 2014). There are many that feel that the effectiveness of integration with science and computer science is understudied (Grover & Pea, 2013; Voogt et al., 2015).

The use of visual programming languages like Scratch, StarLogo Nova and Alice have been seen as tools to develop the interests of young learners in CS and CT as well as increasing the motivation of students and their general learning outcomes (Resnick, 2013). Lewis and Shah (2012) showed that Scratch programming was highly correlated to increased math scores. Calao, Moreno-León, Correa, and Robles (2015) were also able to show that coding in middle school math classes led to improved understanding of mathematical processes.

Professional development for teachers

The core components of teacher PD in CS typically focus on increasing teachers' knowledge around computer science concepts (Barr & Stephenson, 2011). It is important to increase the self-efficacy through content knowledge of the teachers which will in turn have an impact on their instructional practices (Ekmekci, Parr, & Fisher, 2018; Garet, Porter, Desimone, Birman, & Yoon, 2001). In terms of PD, scaffolded PD was significantly superior to PD through self-study in terms of teacher beliefs and motivation, instructional quality, and student achievement (Kleickmann, Tröbst, Jonen, Vehmeyer, & Möller, 2016). PD effects on student learning were mediated only slightly by teacher beliefs. However, teachers' instructional practice emerged as a substantial mediator of PD effects on student achievement (Kleickmann et al., 2016). Providing scaffolded support increases the success of implementations (Lee et al., 2011; Lee, Psaila Dombrowski, & Angel, 2017). Although there has been an emphasis on increasing the content knowledge of teachers in computer science concepts, this does not sufficiently address why there are differences in implementation.

Conclusion

CT and CS education are dynamic areas of research, with many avenues to explore and many gaps to address, especially when these concepts are integrated into STEM classes. Despite efforts from a number of curriculum designers and education researchers, there is insufficient

information on effective models for integration of CS/CT across content areas. Initial data shows that integration is an effective way for students to learn both the content knowledge as well as CS and CT however more work is needed to understand which STEM concepts are more supported by which CS or CT strategies. Teacher professional development models have mainly focused on improving teacher content knowledge in CS rather than the integration of CS/CT into their content area. Ongoing teacher support is needed during the academic year to allow for teachers to effectively implement CS/CT curriculum. There are few professional development models that simultaneously address STEM content knowledge along with CS/CT.

References

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832-835.

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.

Blikstein, P. (2012). *Bifocal modeling: a study on the learning outcomes of comparing physical and computational models linked in real time*. Paper presented at the Proceedings of the 14th ACM international conference on Multimodal interaction, Santa Monica, California, USA.

Brackmann, C. P., Rom, M., #225, n-Gonz, #225, lez, . . . Barone, D. (2017). *Development of Computational Thinking Skills through Unplugged Activities in Primary School*. Paper presented at the Proceedings of the 12th Workshop on Primary and Secondary Computing Education, Nijmegen, Netherlands.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking.* Paper presented at the Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada.

Calao, L. A., Moreno-León, J., Correa, H. E., & Robles, G. (2015). Developing mathematical thinking with scratch. In *Design for teaching and learning in a networked world* (pp. 17-27): Springer.

CSTA, I. (2011). Computational thinking. Teacher resources. . Retrieved from http://csta.acm.org/Curriculum/sub/CurrFiles/472.11CTTeacherResources_2ed-SP-vF.pdf.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education, 58*(1), 240-249.

DiSessa, A. A. (2001). *Changing minds: Computers, learning, and literacy*: Mit Press.

Ekmekci, A., Parr, R., & Fisher, A. (2018). *Results from Rice University WeTeach_CS: A Computer Science Teaching Collaborative Serving Teachers with Different Needs through Variety of Pathways.* Paper presented at the Society for Information Technology & Teacher Education International Conference.

Gallup. (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

Garet, M. S., Porter, A. C., Desimone, L., Birman, B. F., & Yoon, K. S. (2001). What makes professional development effective? Results from a national sample of teachers. *American educational research journal, 38*(4), 915-945.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher, 42*(1), 38-43.

Grover, S., Pea, R., & Stephen, C. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199-237.

Guzdial, M. (1994). Software-realized scaffolding to facilitate programming for science learning. *Interactive Learning Environments, 4*(1), 001-044.

Guzdial, M. (2008). Education Paving the way for computational thinking. *Communications of the ACM, 51*(8), 25-27.

Kafai, Y. B., Peppler, K. A., & Chiu, G. M. (2007). High tech programmers in low-income communities: Creating a computer culture in a community technology center. In *Communities and technologies 2007* (pp. 545-563): Springer.

Kafai, Y. B., & Resnick, M. (1996). *Constructionism in practice: Designing, thinking, and learning in a digital world*: Routledge.

Kleickmann, T., Tröbst, S., Jonen, A., Vehmeyer, J., & Möller, K. (2016). The effects of expert scaffolding in elementary science professional development on teachers' beliefs and motivations, instructional practices, and student achievement. *Journal of Educational Psychology, 108*(1), 21.

Lead States, N. (2013). Next generation science standards: For states, by states. In: The National Academies Press Washington, DC.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., . . . Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32-37. doi:10.1145/1929887.1929902

Lee, I., Psaila Dombrowski, M., & Angel, E. (2017). *Preparing STEM Teachers to offer New Mexico Computer Science for All.* Paper presented at the 48th ACM Technical Symposium on Computing Science Education, Seattle, Washington, USA.

Lewis, C. M., & Shah, N. (2012). *Building upon and enriching grade four mathematics standards with programming curriculum.* Paper presented at the Proceedings of the 43rd ACM technical symposium on Computer Science Education.

National Research Council. (2010). *The Scope and Nature of Computational Thinking*. Retrieved from Washington, D.C.:

Orton, K., Weintrop, D., Beheshti, E., Horn, M., Jona, K., & Wilensky, U. Bringing Computational Thinking Into High School Mathematics and Science Classrooms. *Transforming Learning, Empowering Learners*.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*: Basic Books, Inc.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning, 1*(1), 95-123.

Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism, 36*(2), 1-11.

Repenning, A., & Ioannidou, A. (2008). *Broadening participation through scalable game design.* Paper presented at the ACM SIGCSE Bulletin.

Resnick, M. (2013). Learn to code, code to learn. *EdSurge, May, 54*.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Silverman, B. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60-67.

Schanzer, E., Fisler, K., Krishnamurthi, S., & Felleisen, M. (2015). *Transferring skills at solving word problems from computing to algebra through bootstrap.* Paper presented at the Proceedings of the 46th ACM Technical symposium on computer science education.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies, 20*(4), 715-728.

Wagh, A., Cook-Whitt, K., & Wilensky, U. (2017). Bridging inquiry-based science and constructionism: Exploring the alignment between students tinkering with code of computational models and goals of inquiry. *Journal of Research in Science Teaching*.

Webb, D. C., Repenning, A., & Koh, K. H. (2012). *Toward an emergent theory of broadening participation in computer science education*. Paper presented at the Proceedings of the 43rd ACM technical symposium on Computer Science Education, Raleigh, North Carolina, USA. http://delivery.acm.org/10.1145/2160000/2157191/p173-webb.pdf?ip=64.106.111.2&id=2157191&acc=ACTIVE%20SERVICE&key=B63ACEF81C6334F5%2E1447575D8884B3D4%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=720483329&CFTOKEN=29890755&__acm__=1485297960_b916c414e21769a21d104d044440a6f9

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology, 25*(1), 127-147. doi:10.1007/s10956-015-9581-5

Wilensky, U., Brady, C. E., & Horn, M. S. (2014). Fostering computational literacy in science classrooms. *Communications of the ACM, 57*(8), 24-28.

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

Wing, J. M. (2011). Research notebook: Computational thinking—What and why? . *The Link Magazine*.

Yaşar, O. (2017). The essence of computational thinking. *Computing in Science & Engineering, 19*(4), 74-82.