Exploring the Prospects of Knowledge Building in Advancing Computational Thinking

Dina Soliman, University of Toronto, dinaa.soliman@utoronto.ca

Abstract: This paper explores prospects of Knowledge Building in advancing computational thinking by drawing upon commonalities between these fields and opening new possibilities for enhancing discourse surrounding ideas and artifacts and for assessing levels of computational thinking. Toward this end I review definitions of computational thinking, particularly in the context of K-12 education. Factors of significance for integration into Knowledge Building include discourse to generate, extend, and explore ideas and artifacts from a computation perspective and use of new forms of assessment so that computational thinking is integral to day-to-day knowledge work. To determine the first level of computational thinking three Knowledge Forum databases of an elementary school were analyzed for the use of computational thinking vocabulary prior to any committed time to advance computational thinking. As expected, results showed very little use of computational keywords. Next design iterations will facilitate discourse surrounding computational ideas and artefacts to facilitate an advanced level of computational thinking involving computational tools as mediums of expression, connecting, and questioning. In addition, as part of a global *Saving the Planet, Saving lives* initiative, students' work will be connected to a broader network of communities while working on parallel issues to support sustained creative work with ideas.

Introduction

Worldwide attention to computational thinking can arguably be traced to Jeanette Wing's (2006) influential article where she claims that computational thinking is a "universally applicable attitude and skill set" (p.33) that should be added to every child's analytical ability. Debates that followed in the academic community focused on its definition and means, with a lack of consensus on how it should be taught and evaluated (Barr & Stephenson, 2011; Brennan & Resnick, 2012; Denning, 2017; Grover & Pea, 2013a). In this paper, I outline literature that defines computational thinking and the different attempts made to bring computational thinking to K-12 education. I discuss approaches to assess computational thinking in the context of visual programming. Finally, I highlight the prospects of Knowledge Building as a pedagogical approach to advance computational thinking.

Background

The roots of computational thinking can be traced to the work of computing pioneers like Alan Perlis who claimed that computing would eventually automate all processes and 'algorithmic thinking' will pervade all fields (Denning, 2017; Guzdial, 2008). Seymour Papert (1980) extended this vision in his book Mindstorms in which he recognized the benefits of teaching students how to program and argued that working with computers can influence how people think. Papert emphasized the role of computer modeling in learning and argued that procedural thinking can help children form concrete forms to areas of knowledge previously considered too abstract.

A resurgence of interest in computational thinking followed the publication of Wing's seminal article "Computational Thinking", where she describes computational thinking as an approach to solve problems, design systems, and understand human behaviour by drawing on key concepts in computer science (Wing, 2006). In a more refined definition, Wing accepts that computational thinking is "*the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out*" (Cuny et al., 2010, p.1). Alfred Aho (2012) echoes Wing's emphasis on considering computational thinking a thought process but emphasizes it should be used in conjunction with a clearly defined model that fits with the problem being investigated. Aho defines computational thinking as *"the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms"*(p.832). When a model is not available, computational thinking becomes a research activity that involves devising new models to formulate and solve problems.

Computational Thinking in K-12: Definitions and Frameworks

The renewed emphasis on computational thinking catalyzed by Wing's article prompted a number of organizations and researchers to propose an operational definition for computational thinking in K-12 (Barr & Stephenson, 2011; Denning, 2017; Grover & Pea, 2013a; Wing, 2008). In 2010, the National Research Council conducted two workshops to explore the nature and scope of computational thinking and its educational implications based on Wing's definition (NRC, 2010; NRC, 2011). In the first workshop, participants agreed that abstraction is the

keystone of computational thinking but there was no consensus on how computational thinking should be defined. Discussions in the second workshop that focused on pedagogy also showed lack of agreement on the best approaches to teach computational thinking (Grover & Pea, 2013a).

Another project co-led by the International Society for Technology in Education (ISTE) and the Computer Science Teacher Association (CSTA) aimed to provide a framework to help educators bring computational thinking to different subjects in K-12. The efforts resulted in an operational definition of computational thinking that describes it as a problem-solving process that involves problem formulation, analysis and organization, use of models and simulation, algorithmic thinking, automation, thinking of and testing alternative solutions, and generalizing the problem-solving process to be used across domains. They determined that the ability to persist in solving problems, to tolerate complex, ambiguous, and open-ended problems, and to collaborate with others towards a shared goal are essential attitudes that enhance computational thinking skills (Barr et al., 2011). Based on these outcomes, Barr and Stephenson (2011) produced a structured checklist that identified core computational thinking concepts and suggested examples of how they might be embedded in different subjects. Another notable effort made by the Royal Society (2012) in the UK produced a more precise definition for computational thinking as the *"process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes*" (p.29).

Selby and Woollard (2013) highlight the necessity of a robust definition of computational thinking to facilitate proper curriculum design and assessment. Their effort to collate common concepts that appeared in literature determined that computational thinking is a thought process that reflects the ability to systematically apply abstraction, decomposition algorithmic thinking, evaluation, and generalization to provide automated solutions. Weintrop et al. (2016) argue that embedding computational thinking in math and science disciplines will give learners a more realistic understanding of the fields and provide them with a deeper content learning experience. They define computational thinking in terms a taxonomy composed of four main categories of practices, including data, modeling and simulation, computational problem solving, and systems thinking. Kotsopoulos et al. (2017) propose a more pedagogically oriented framework for introducing computational thinking that consists of unplugged, tinkering, making/constructing, and remixing activities.

Computational Thinking, Creativity, and Knowledge Construction

Most definitions of computational thinking seem to focus on skills and attitudes required to solve complex problems where the use of technology is key (e.g. Barr & Stephenson, 2011; Grover & Pea, 2013a; Weintrop et al., 2016). A broader perspective suggests that computational thinking can foster creativity by allowing students to build tools that can have a major impact on society. Mishra and Yadav (2013) claim that computational thinking facilitates new forms of expression and argue that combining computational thinking with deep knowledge of a discipline can result in creative solutions otherwise not possible. Romero et al. (2017) argue that creative programming activities that present learners with ill-defined problems can be used to foster knowledge building through orchestrating and enhancing creative activities. More recently, Paniagua and Istance (2018) classify computational thinking as an innovative pedagogy which can facilitate the development of transversal skills and promote creativity and meaningful learning. While they acknowledge the importance of coding and algorithmic thinking, they argue that engaging students in knowledge creation is key to the process;

> "*computational thinking is sometimes seen as 'algorithmic thinking' and writing codes, but if the goal is to learn how to look to at a problem in new ways, or to find new answers according to a given sets of possibilities - as a computer or programming language might -, then the skills associated with the generation of ideas and openness to develop and explore ideas need to be practised along with computational thinking skills."* (p.107)

This view is corroborated by ISTE standards which state that students should be computational thinkers and knowledge constructors if they are to succeed in a constantly evolving technological world. As knowledge constructors, students should build knowledge by "*actively exploring real-world issues and problems, developing ideas and theories and pursuing answers and solutions*." (ISTE, 2016, p.1).

Computational Thinking through Programming

Over half a century ago Perlis anticipated what is happening today: machine automation is changing operations across all disciplines (Gudzial, 2008). Perlis's call to teach programming to everyone was elaborated by Papert (1980) whose work on developing the Logo programming language was accompanied by a vision that it is the child who should program the computer and not vice versa. In doing so, the child "*both acquires a sense of mastery*

over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building" (p.5).

While the attempts outlined earlier consider computational thinking a *thought* or *problem-solving* process, Brennan and Resnick (2012) propose a framework that adopts a constructionist approach to learning (Harel & Papert, 1991) which highlights that learning takes place by engaging learners in the design and development of artifacts. The framework is based on Resnick's work on Scratch, a visual programming environment that enables children to learn programming by creating, sharing, and collaborating on interactive and meaningful projects (Maloney et al., 2010). The framework is used to determine how programming activities on Scratch can support the development of computational thinking in young learners in terms of three dimensions. First, *computational concept*s that learners engage in while programming, such as loops, conditions, and operators. Second, *computational practices* that learners develop as they work with different concepts, which include being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing. Third, *computational perspectives* that learners develop about themselves and about their worlds, which includes their ability to express ideas, connect with others in their community, and question their experiences in the technological world and respond to these questions via design (Brennan & Resnick, 2012).

Potentials and Challenges

Despite calls for alternative views to introduce computational thinking in K-12, current research shows that teaching visual programming results in substantial gains in students' capacity to think computationally (Lye & Koh, 2014; Zhang & Nouri, 2019). This may be due to the proximity of visual programming representations to human language which reduces the cognitive load on novice programmers and allows them to focus the structures and logic of their program (Kelleher & Pausch, 2005; Maloney et al., 2010). Tools like Scratch can help students think computationally where the ultimate goal is not simply to code but to produce artefacts that genuinely interests them (Maloney et al., 2010). For example, a study conducted by Lee (2010) found that Scratch enabled primary school students to learn programming while creating multi-media language-arts projects. Another study conducted by Burke (2012) on middle school children learning languages via Scratch suggests that digital storytelling can effectively facilitate learning composition and at the same time introduce students to coding at early ages.

While it is unreasonable to expect all students to become programming experts, some researchers argue that basic programming is an important 21st century competency (Barr & Stephenson, 2011; Grover & Pea, 2013a; Weintrop et al., 2016). Others such as Shein (2014) assert that not all students need coding skills although learning how to think like a programmer can be helpful in many areas. Shein argues that teaching programming views coding as the goal which puts '*the method before the problem'*. Grover and Pea (2018) share this concern and argue that problem formulation is key in the problem-solving process, which can be taught without programming or computers. Other views question whether computational thinking, through programming or otherwise, is indeed as important as promoted by advocates. Hemmendinger (2010) believes the benefits of computational thinking outlined by Wing are overstated and "scarcely peculiar to computing" (p.4). He suggests focusing less on computational *thinking* and more on computational *doing* - figuring out how to use computational tools to accomplish work in new ways. Denning (2017) agrees that recent definitions of computational thinking made overreaching and unsubstantiated claims and expectations, leaving teachers confused on how they should teach or assess computational thinking. Easterbrook (2014) criticizes the overly simplistic notion of computational thinking as a technology-driven problem-solving approach, and proposes systems thinking as a more sustainable approach that considers the interrelationships between technology, human behavior, and environmental impacts.

Assessment

Measuring students' computational abilities remains a primary challenge that teachers and educational researchers still struggle with. Denning (2017) suggests directly testing for competencies rather than knowledge, but provides no direction or examples of how to effectively assess the development of computational thinking skills. While there have been some attempts to develop tools for assessing computational thinking (for e.g. Chen et al., 2017; Zhong et al., 2016), it seems that each tool will only provide a partial view to students' computational thinking (Román-González et al., 2019). Brennan and Resnick (2012) propose three approaches for assessing their framework: Scratch project portfolio analysis, artifact-based interviews, and discussions around design scenarios. However, they acknowledge the limitations of these approaches and call for multiple and alternative means of assessment, highlighting the importance of reflection and discourse about the produced artefacts. A review of empirical research conducted by Lye and Koh (2014) on teaching and learning computational thinking through programming showed dearth of research, with only 9 peer-reviewed studies conducted in K-12, mostly in after-school settings. Moreover,

most studies reviewed explored the *computational concepts* dimension of the framework, although the other two dimensions - *computational practices and perspectives* – have become more relevant as the basis for introducing computational thinking in K-12, as they pertain to the common idea of computational thinking as a transferable, problem-solving thought process (Aho, 2012; Barr et al., 2011; Cuny et al., 2010). To examine computational practices and perspectives, Lye and Koh (2014) suggest capturing students' programming processes by recording on-screen activities and engaging students in think-aloud protocols. A more recent review by Zhang and Nouri (2019) still revealed that most studies examined computational thinking *concepts* with fewer studies exploring *practices* and *perspectives,* and reemphasized Lye and Koh's (2014) call for more scholarly research that explores these two dimensions. In this spirit, I address Knowledge Building as carrying potential for advancing computational thinking.

Knowledge Building to Promote Computational Thinking

Knowledge Building is a principle-based, educational approach developed by Scardamalia and Bereiter, both pioneers of computer-supported collaborative learning (CSCL). The Knowledge Building approach focuses on the growing need for the ability to work creatively with knowledge. The essence of Knowledge Building is the "*production and continual improvement of ideas of value to a community*" through shared discourse that aims to advance the knowledge and expertise of the community (Scardamalia & Bereiter, 2003, p.1371). The concept Knowledge Building as the collaborative creation and advancement of public knowledge can be considered in the light of six themes: 1) community rather than individual knowledge advancement, 2) idea improvement rather than correct/incorrect ideas, 3) knowledge *of* (deep knowledge) rather than knowledge *about*, 4) discourse as collaborative problem solving rather than argumentation, 5) constructive use of authoritative resources, and 6) understanding as emergent (Scardamalia & Bereiter, 2006). Knowledge Building discourse is facilitated by the Knowledge Forum technology that is especially designed to support and promote advanced knowledge work (Scardamalia, 2004).

Teacher reports suggest that students engaged in Knowledge Building surpass expectations with respect to creativity, teamwork, problem solving, and other indicators known as 21st century competencies (Khanlari et al., 2019). Research indicates that students engaged in Knowledge Building experienced gains in core discipline knowledge (for e.g., Chan & van Aalst, 2003; Zhang et al., 2011) which Mishra and Yadav (2013) deemed as necessary if computational thinking is to result in the production of creative technology-enhanced solutions. Moreover, evidence indicates that Knowledge Building advances a range of literacies and skills such as collaboration, argumentation, digital literacies, and problem-solving skills (for e.g. Gan et al., 2010; Zhang et al., 2007), which were highlighted in the literature as core computational thinking competencies. The use of computational vocabulary is also noted in computational thinking literature as a core capability that creates awareness of computational practices and promotes a classroom culture conducive to computational thinking (e.g. Barr & Stephenson, 2011; Lu & Fletcher, 2009), but so far there is a dearth of research investigating practices to promote the use of computational thinking vocabulary. Knowledge Building research shows that discourse were students discuss and build-on each other's ideas and experiments results in significant growth of students' productive academic, domain-specific, and cross-domain vocabulary (Hong et al., 2015; Khanlari et al., 2019; Sun et al., 2008).

Lye and Koh (2014) argue that proper guidance on the cognitive aspects of the computational *practices* and *perspectives* is necessary for a proper educational experience. They suggest that educational environments that foster computational thinking should be designed to promote authentic problem solving. That is, students should work on problems relevant to them, which is emphasized in the Knowledge Building principle *Real Ideas, Authentic Problems*. Environments should also allow teachers to scaffold the process of problem-solving that enables learners to make their thinking process more explicit, particularly while working on programming projects (Lye & Koh, 2014). An important feature in Knowledge Forum is the *Scaffold*, which can represent different epistemological terms such as *I need to understand* or *putting our thoughts together* that can help frame student's thinking and allow them to express and reflect on their ideas. Scaffolds can be customized to support the particular discourse needs and can be searched and analyzed to determine patterns of use and contribution to the advancement of student and community knowledge (Scardamalia, 2004).

Grover and Pea (2013a) argue that there is a scarcity of studies that consider contemporary research in the learning sciences in areas like socio-cultural learning, and activity, interaction, and discourse analyses. Realizing this gap, the authors conducted a one-day workshop that introduced middle-school students to programming concepts via a discourse-intensive pedagogy. The curriculum was designed around engaging students in "*knowledge building discussions in concert with engaging in computationally rich activities*" (Grover & Pea, 2013b, p.724). Results show that instructor and participants' responses to emergent issues triggered opportunities for introducing more advanced computational concepts and resulted in a significant growth in computational vocabulary. Despite being an exploratory study, the findings support the idea that Knowledge Building discourse around programming projects can result in

significant advancements of computational thinking vocabulary and competencies. Moreover, examining students' Knowledge Building discourse and augmenting the findings with results of analytical tools that evaluate programming projects (such as Dr. Scratch) can offer multiple means of assessing computational learning and help provide a more holistic view of student computational understanding (Brennan & Resnick, 2012; Grover, Cooper, & Pea, 2014).

Knowledge Building, Programming, and Papert's Vision

Papert warned against taking a technocentric perspective to programming; the centrality should not be on the technical object, but rather on the culture. According to Papert (1987), technocentric approaches "*often betray a tendency to think of 'computers' and of 'Logo' as agents that act directly on thinking and learning; they betray a tendency to reduce what are really the most important components of educational situations—people and cultures—to a secondary, facilitating role*." (p 23). This technocentric view is perhaps what stood in the way of Papert's vision of a world where children do not just interact with technology but are able to create their own models and express their ideas using technology. As highlighted by Resnick (2012), most teachers today see programming as a "narrow, technical activity, appropriate for only a small segment of the population" (p.42). Hence, reviving Papert's vision requires multi-faceted efforts to bring about technologies that are more social and meaningful with the potential to fundamentally change educational practice. Visual programming tools such as Scratch are considerable leaps towards this direction by providing platforms where learners can create personally meaningful and shareable projects (Resnick, 2012). However, while such tools simplify the acquisition of cognitive skills, primary focus is not on empowering student discourse about their ideas. Students should be able to use code for creative self-expression and to solve problems that matter to them in an environment that promotes student agency where new knowledge emerges out of social interactions. Hence, a combined, interdisciplinary approach to teaching computational thinking where the primacy is to improvable ideas and facilitating students' epistemic agency can help realize this vision. In this setup, students can engage in Knowledge Building discourse where multiple perspectives and critiques are encouraged, not only about technical aspects of digital artefacts but also with how these artefacts connect with other areas of their lives. Knowledge Building discourse can be facilitated using Knowledge Forum technology, where scaffolds can be customized and used "opportunistically and flexibly" (Scardamalia, 2004, p.189) to guide student inquiry and thinking in multiple directions while working on designing and creating programming projects.

Use of Computational Thinking Vocabulary in Knowledge Building Environments

Tsoukas (2009) explains that dialogue "aims at removing some kind of unsettledness (or perplexity) experienced by the parties involved, through their reasoning together by verbal exchanges." (p.943). Specifically, productive dialogue takes place when participants find a *common languag*e. Hence, for students to engage in productive computational thinking discourse and be able to communicate their computational ideas more effectively a common computational thinking vocabulary should be established. As argued by Lu and Fletcher (2009), a Computational Thinking Language consisting of vocabularies that capture core computational thinking concepts like abstraction and information should permeate pedagogy.

As a starting point to research that explores the prospects of Knowledge Building in advancing computational thinking, I searched three Knowledge Forum databases of an elementary school for occurrences of certain computational thinking keywords and their common derivatives as students engaged in Knowledge Building discourse in different science topics (see Table 1). Results show that in all three databases students barely used any keywords (see Table 2, Table 3, and Table 4).

Table 1 Computational thinking keywords

Algorithm	Conditional	Input	Output	Initialize	Variable
Boolean	Sequencing	Branching	Iteration	Loop	Abstraction
Debug	Debugging	Binary	Data	Automation	Decomposition
Function	Parameter	Analysis	Simulation	Parallelization	Program
Testing	Pattern	Coding			

Table 2 Computational thinking keywords (2017-2018 database)

Keyword	Number of occurrences
Data	2
Programs	1

Patterns	1

Table 3 Computational thinking keywords (2018-2019 database)

Keyword	Number of occurrences
decompose	2
Testing	1

Table 4 Computational thinking keywords (2019-2020 database)

Keyword	Number of occurrences
Data	16
decompose	1
Program	1
Testing	1

A closer look at the notes where the keywords occurred revealed that they were not used within a computational thinking context, except for one note in the 2017/2018 database where the student used both 'patterns' and 'data' in the same note: "… *If we can figure out some of the trends and patterns of the past maybe we can use that information and data to do predict the climate in the future*". This is not entirely surprising as the school does not teach computational thinking or integrate computational thinking concepts into the curriculum of other subjects.

Evidence from Grover and Pea's pilot study (2013b) of introducing computational thinking using discourse-intensive pedagogy suggests that learning environments may be "consciously designed for computational discourse to help children develop a vocabulary that is faithful to CS as a discipline" (p.727). My own viewpoint for advancing computational thinking is based on designing two main interventions. The first is to introduce students to computational thinking concepts via computational *Knowledge Building* discourse. In their computational thinking class students will propose new projects and discuss implementation interspersed with work on Scratch. The belief and goal are that engaging students in face-to-face Knowledge Building talk as well as Knowledge Forum discourse will support them while working on programming projects and will help them acquire transferable computational thinking competencies and develop a computational thinking vocabulary. To facilitate computational thinking discourse, Knowledge Forum scaffolds can be created and adjusted for different grade levels to inspire student generation of programmable ideas (for e.g. this program could be improved by…, a new pattern..., a better sequence…). The second stage involves leveraging the acquired computational thinking competencies to engage students in the co-creation of digital artefacts (for e.g. simulations, digital stories) that address real problems tackled while engaging in science Knowledge Building discourse. Scaffolds can be created to facilitate the idea of 'doing science' through programming (for e.g. such "we need a simulation to show…, this model would clarify). The expectation is more sustained, productive, and integrative use of computational thinking vocabulary, concepts, and practices over time, and the development of students' perspective for programming as a tool to create artefacts that are useful and valuable to their communities.

Future Research

This initial, benchmark-establishing research involved analysis of three Knowledge Forum databases of an elementary school for the use of computational thinking vocabulary over a 3-year time period. The goal was to take a snapshot, prior to any direct engagement in computational thinking, to determine use in unsupported contexts. Next stages will involve design iterations, using scaffolds and visualization of computational concepts to engage students more productively in discourse surrounding computational ideas and artifacts, with Brennan and Resnick's dimensions as the guiding framework. Students will use Knowledge Forum for planning, discussion, and evaluation of the games, animations, simulations and models they create using Scratch. In addition, as part of a global *Saving the Planet, Saving lives* initiative, students' work will be connected to a broader network of communities while working on parallel issues to support sustained creative work with ideas. A "levels of computational thinking" analysis will be developed, extending from limited use of terms and use without computational implications evident to seeing and using computational tools as mediums of expression, connecting, and questioning.

Conclusion

The aim of this paper is to shed light on different attempts to define and facilitate computational thinking in K-12 to demonstrate the prospects of Knowledge Building as a promising approach to advance computational thinking. Engaging students in Knowledge Building discourse, with the use of appropriate scaffolds to guide, promote, and capture students' ideas and thought processes while working on programming projects is likely to contribute positively to advancing computational thinking. Factors of significance for integration into Knowledge Building include discourse to generate, extend, and explore ideas and artefacts from a computational perspective and use of new forms of assessment so that computational thinking is integral to day to day knowledge work. Research in this area will be a valuable contribution to the field of computational thinking education as well as the Knowledge Building literature.

References

Aho, A.(2012). Computation and Computational Thinking. *The Computer Journal,* 55(7). https://doi.org/10.1093/comjnl/bxs074.

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. Learning& Leading with Technology, 38(6), 20-23.

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. https//doi.org/: 10.1145/1929887.1929905.

Bereiter, C., & Scardamalia, M. (2003). Learning to work creatively with knowledge. In E. De Corte, L. Verschaffel, N. Entwistle, & J. van Merriënboer (Eds.), *Powerful learning environments: Unraveling basic components and dimensions (Advances in Learning and Instruction Series)* (pp. 55-68). Oxford, UK: Elsevier Science.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual American Educational Research Association Meeting,* Vancouver, BC, Canada.

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. The Journal of the Learning Sciences, 2(2), 141–178.

Burke, Q. (2012). The markings of a new pencil: Introducing programming-as writing in the middle school classroom. *Journal of Media Literacy Education*, 4(2), 121–135.

Chan, C., & Van Aalst, J. (2004). Learning, assessment and collaboration in computer supported environments. In J. W. Strijbos, P. A. Kirschner & R. L. Martens (Eds.), *What we know about CSCL*, (pp. 87—112). Netherlands: Springer.

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, 109, 162–175. https://doi.org/10.1016/j.compedu.2017.03.001.

Collins, A. (1992). Towards a design science of education. In E. Scanlon & T. O'Shea (Eds.), New Directions in Educational Technology (pp. 15-22). Berlin: Springer-Verlag.

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. Unpublished manuscript, referenced in http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Denning, P. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, 60(6), 33-39.

Easterbrook. (2014). From Computational Thinking to Systems Thinking: A conceptual toolkit for sustainability computing. *Proceedings of the 2014 conference Icomputational thinking for Sustainability (*pp 235-244). Atlantis Press. https://doi.org/10.2991/ict4s-14.2014.28.

Gan, Y., Scardamalia, M., Hong, H., & Zhang, J. (2007) Making thinking visible: Growth in graphical literacy, grades 3 to 4. *Canadian Journal of Learning and Technology*, 36(1).

Grover, S., & Pea, R. (2013a). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Grover, S., & Pea, R. (2013b). Using a discourse-intensive pedagogy and android's app inventor for introducing computational concepts to middle school students. *Proceeding of the 44th ACM technical symposium on Computer science education*, Denver, Colorado, USA, 723-728.

Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12. In *Proceedings of the 2014 conference on Innovation & technology in computer science education (ITiCSE '14)*. Association for Computing Machinery, New York, NY, USA, 57–62. https://doi.org/10.1145/2591708.2591713

Grover, S., & Pea, R. (2018). Computational Thinking: A competency whose time has come. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer Science Education: Perspectives on Teaching and Learning*. London: Bloomsbury.

Guzdial, M. (2008). Education Paving the way for computational thinking. *Communications of the ACM*, 51, 25-27. https://doi.org/10.1145/1378704.1378713.

Harel, I., & Papert, S. (Eds.). (1991). Constructionism. Ablex Publishing.

Hemmendinger, D.(2010). A plea for modesty. *ACM Inroads* 1(2), 4–7. https://doi.org/10.1145/1805724.1805725

Hong, H., Scardamalia, M., Messina, R., & Teo, C.(2015). Fostering sustained idea improvement with principle-based knowledge building analytic tools. Computers & Education , 89 , 91–102. http://dx.doi.org/10.1016/j.compedu.2015.08.012'

International Society for Technology in education. ISTE Standards for Students. 2016. Retrieved from https://www.iste.org/standards/for-students.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, 37(2), 83–137. https://doi.org/10.1145/1089733.1089734.

Khanlari, A., Zhu, G., & Scardamalia, M. (2019). Knowledge Building Analytics to Explore Crossing Disciplinary and Grade-Level Boundaries. *Journal of Learning Analytics* 6 (3), 60-75.

Kotsopoulos, D., Floyd, L., Khan, S., Namukasa, I.K., Somanath, S., Weber, J., & Yiu, C. (2017). A pedagogical framework for computational thinking. *Digital Experiences in Mathematics Education*, 3, 144-171. https://doi.org/10.1007/s40751-017-0031-2.

Lee, Y.-J. (2010). Developing computer programming concepts and skills via technology-enriched language-art projects: A case study. *Journal of Educational Multimedia and Hypermedia*, 19(3), 307–326.

Lu, J., & Fletcher, G. (2009). Thinking about computational thinking. In *Proceedings of the 40th ACM technical symposium on Computer science education (SIGCSE '09)*. Association for Computing Machinery, New York, NY, USA, 260–264. https://doi.org/10.1145/1508865.1508959.

Lye, S., & Koh, J. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Maloney, J., Resnick, M., Rusk, N., Silverman, B. & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education,* 10 (4). https://doi.org/10.1145/1868358.1868363.

Mishra, P., & Yadav, A. (2013). Of art and algorithms: Rethinking technology & creativity in the 21st century. *TechTrends*, 57(3), 10-14.

National Research Council. (2010). *Committee for the Workshops on Computational Thinking: Report of a workshop on the scope and nature of computational thinking*. Washington, DC: National Academies Press.

National Research Council. (2011). *Committee for the Workshops on Computational Thinking: Report of a workshop of pedagogical aspects of computational thinking*. Washington, DC: National Academies Press.

Paniagua, A. and Istance, D. (2018). *Teachers as Designers of Learning Environments: The Importance of Innovative Pedagogies*, Educational Research and Innovation, OECD Publishing, Paris, https://doi.org/10.1787/9789264085374-en.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*; Basic Books: New York, NY, USA, 1980; ISBN 978-0-465-04627-0

Papert, S. (1987). Information Technology and Education: Computer Criticism vs. Technocentric Thinking. Educational Researcher, 16(1), 22–30. https://doi.org/10.3102/0013189X016001022

Resendes, M., & Chuy, M. (2010). Knowledge building for historical reasoning in Grade 4. In K.Gomez, L. Lyons & J. Radinsky (Eds.), *Proceedings of the 9th International Conference of the Learning Sciences* (ICLS '10), Vol. 2 (pp. 443- 444). International Society of the Learning Sciences, Chicago, USA.

Resnick, M. (2012). Reviving Papert's Dream. *Educational Technology, 52*(4), 42-46.

Román-González M., Moreno-León J., Robles G. (2019) Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions. In: Kong SC., Abelson H. (Eds) *Computational Thinking Education*. Springer, Singapore.

Romero, M., Lepage, A. & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*, 14(42). https://doi.org/10.1186/s41239-017-0080-z.

Royal Society. (2012). Shut down or restart: The way forward for computing in UK schools. Retrieved from https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

Scardamalia, M. (2004). CSILE/Knowledge Forum®. In *Education and technology: An encyclopedia* (pp. 183-192). Santa Barbara: ABC-CLIO.

Scardamalia, M., & Bereiter, C. (2003). Knowledge Building. In Encyclopedia of Education. (2nd ed., pp. 1370-1373). New York: Macmillan Reference, USA.

Scardamalia, M., & Bereiter, C. (2006). Knowledge Building: Theory, Pedagogy, and Technology. In R. K. Sawyer (Ed.), *The Cambridge handbook of: The learning sciences* (p. 97–115). Cambridge University Press.

Shein, E. (2014). Should everybody learn to code? *Communications of the ACM*, 57(2),16-18. https://doi.org/10.1145/2557447.

Selby,C., & Woollard, J. (2013). Computational thinking: the developing definition. University of Southampton (E-prints). Retrieved from https://eprints.soton.ac.uk/356481/.

Sun, Y., Zhang, J., & Scardamalia, M. (2010). Knowledge building and vocabulary growth over two years, Grades 3 and 4. *Instructional Science*, 38(2), 147-171.

Tsoukas, H. (2009). A dialogical approach to the creation of new knowledge in organizations. Organization Science, 20, 941-957.

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. https://doi.org/10.1007/s10956-015-9581-5.

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

Wing, J. (2008). Computational Thinking and Thinking about Computing. *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences,* 366, 3717–3725.

Zhang, J., Hong, H., Scardamalia, M., Teo, C., & Morley, E. (2011). Sustaining knowledge building as a principle-based innovation at an elementary school. The Journal of the Learning Sciences, 20(2), 262–307.

Zhang, J., Scardamalia, M., Lamon, M., Messina, R., & Reeve, R. (2007). Socio-cognitive dynamics of knowledge building in the work of 9- and 10-year-olds. Educational Technology Research and Development, 55, 117–145. http://dx.doi.org.ezp2.lib.umn.edu/10.1007/s11423-006-9019-0

Zhang, L. & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education,* 141, 1-25. https://doi.org/10.1016/j.compedu.2019.103607

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research*, 53(4), 562–590. https://doi.org/10.1177/0735633115608444.