Computational Thinking and Applications for Student Learning and Teacher Learning

Stephen Hadden

Educational Technology and Design, University of Saskatchewan

ETAD 802: Historical and Theoretical Foundations of Educational Technology

Dr. Marguerite Koole

April 12, 2021

Students need a robust and extensive computer-based education. Our schools have become an increasingly technological environment. Students have greater access to computational technology for learning than ever before. Following school, many career and work opportunities are situated within an increasingly technological environment. Computational thinking (CT) is a process that can aid students to use these technology tools more effectively in their educational and potential career pursuits. To undertake learning with CT is to invoke an heuristic problem solving method that incorporates hands-on learning while working towards a solution using the best tools available (Yadav et al., 2017). CT has been and is being integrated into programs and curricula in Canadian provinces and around the world (Caeli & Bundsgaard, 2020; European Commission. Joint Research Centre., 2016; Floyd, 2020; Gülbahar & Kalelioğlu, 2017; Hubwieser et al., 2015). The use of CT methods is growing but necessary shifts in practices and understanding around computers-based education and computational thinking still need to occur for both students and teachers. Proponents of computational thinking see it as a multi-disciplinary practice that will broaden the focus of computer science and mathematics instruction (Grover et al., 2015; Wing & Stanzione, 2016).

Computational thinking in the classroom ensures that students can receive a robust computer-based education, providing educators and students with methods and skills to explore subjects with an evolving understanding of the technical tools and processes available. This paper reviews the concepts of computational thinking, the theorical underpinnings, and areas of application for both students and teachers with a focus on middle years education (grades 6-9).

What is Computational Thinking?

Computational thinking, as an educational theory, is an evolving concept. Computational thinking draws elements from comparable thinking styles such as algorithmic thinking or design thinking, and adds the use of computing technology to arrive at a problem solution (Denning, 2017; Jenson & Droumeva, 2016; Papert, 1980). Computational thinking considers the use of tools, both analog and digital, to determine the best method, or methods, to solve problems (Barr & Stephenson, 2011).

Though the first instance of computational thinking is attributed to Seymour Papert, CT rose to prominence following a 2006 article by Jeannette Wing (Dagienė et al., 2017). Wing's article provided several conceptual ideas regarding computational thinking. Her definitions are cited by authors extensively, and many have added to and expanded on her original definitions, see Figure 1. Generally,

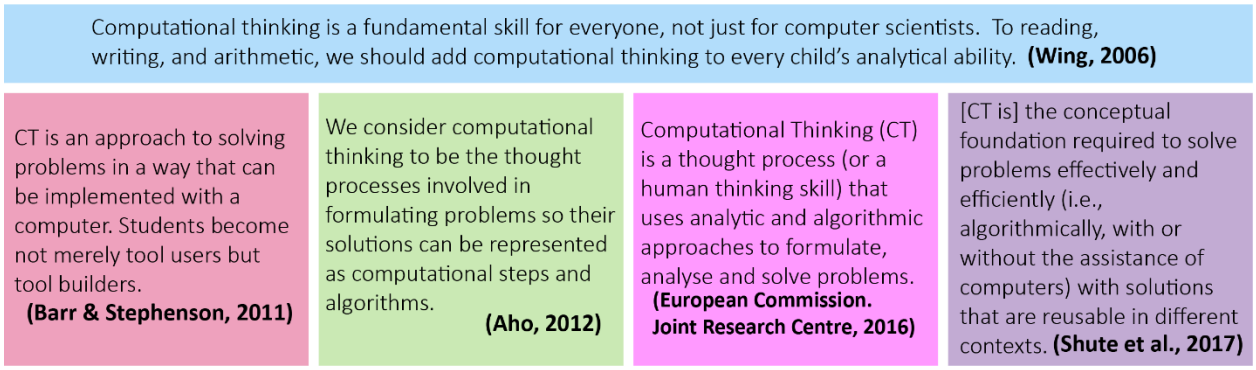

Figure 1: Computational Thinking Definitions

most authors agree that CT is a thinking process that allows students to solve problems using a range of skills, often involving technology. CT, as a process, has avoided codification, and there is some discrepancy as to all that CT entails. A recent study out of Denmark by Caeli and Bundsgaard (2020) stated, "there is still little common understanding of what computational thinking is, how it should be taught, and how it can be assessed". The variety within CT does not negate the potential of computational thinking but reflects upon the flexibility found within the process. As technology changes so can the associated elements of computational thinking (Aho, 2012). Computational thinking evolves as the understanding of computers changes since that understanding is related to how computers are used.

Computational Thinking Skills

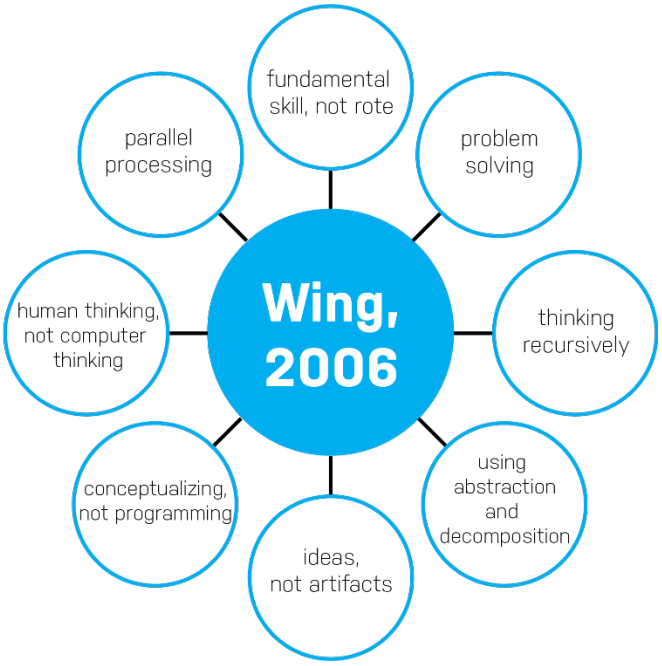

Figure 2: Computational Thinking (Wing, 2006)

Wing (2006) provided an extensive list of computational thinking characteristics that set the foundation for CT, see Figure 2. More recently, the European Commission's Science and Knowledge service released a comprehensive report on the practice and acceptance of CT. The commission report included a list of core skills for computational thinking from various definitions and descriptions. The skills list included concepts of "abstraction, algorithmic thinking, automation, decomposition, debugging, and generalization" (European Commission. Joint Research Centre., 2016), see Table 1. The focus on technology and computational logic extends the capabilities of other forms of thinking such as algorithmic thinking or logical reasoning (Csizmadia et al., 2015). "The power of computational thinking is that it applies to every other type of reasoning. It enables all kinds of things to get done: quantum physics, advanced biology, human computer systems, development of useful computational tools" (Barr & Stephenson, 2011). The skills of CT expand the student's understanding of what is possible through technology, and provides the learner with types of thinking required to explore problems in new ways.

Table 1: Core Skills of Computational Thinking

Abstraction	"Abstraction is the process of making an artefact more understandable through reducing the unnecessary detail" (Csizmadia et al., 2015)
Algorithmic Thinking	"A way of getting to a solution through a clear definition of the steps" (Csizmadia et al., 2015)
Automation	"Process in which a computer is instructed to execute a set of repetitive tasks" (European Commission. Joint Research Centre., 2016)
Decomposition	"A way of thinking about artefacts in terms of their component parts" (Csizmadia et al., 2015)
Debugging	Fixing errors and fine tuning solutions
Generalization	"identifying patterns, similarities and connections, and exploiting those features." (Csizmadia et al., 2015)

The Computational Thinking Process

Many authors have provided frameworks for the computational thinking process. Einhorn (2012) provided a thorough example of the CT process in education,

> "A student, when using programming to tackle a question, has to develop a hypothesis as to how best to solve or answer it, then build, through analysis of the problem, a set of rules (an algorithm) that can be used to test the hypothesis, after which she can review the results (data), and revise the solution. The art of programming requires creativity and inventiveness, logic, algorithmic thinking, and an appreciation of the recursive nature of this process, as the student learns from her failures, refines her work, and gets a deeper understanding of the problem. As with any creation, even once a solution is found – a pattern, an algorithm – the solution can be refined, simplified and beautified, made more elegant."

This description highlights the relationship between CT skills and the application of computer tools in the educational process. It is important to note that the process does not necessarily incorporate all the skills every time, but that there are clear identifiable phases in the process. Wolfram (2020) proposed a 4-step computational thinking process that reasonably summarizes the Einhorn description: 1) define the questions to consider, 2) abstract the process to something computable, 3) compute answers and 4) interpret the results. Palts & Pedaste (2020) provided a cyclical model

for computational thinking similar to both the Einhorn description and Wolfram process, see Figure 3. These models provide structure to encapsulate the skills of the computational thinking process. The process is still flexible but is scaffolded to follow a relatively similar structure when applied.

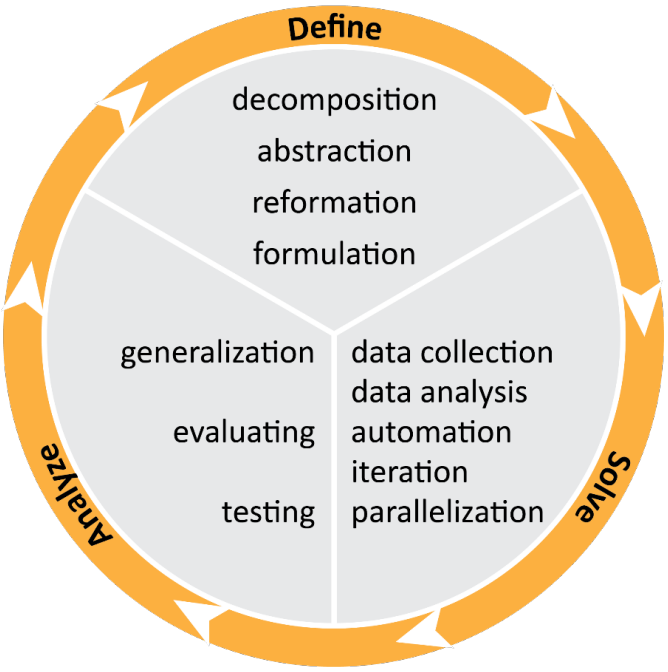

Figure 3: Computational Thinking Cycle: Palts & Pedaste, 2020

The Theoretical Underpinnings of Computational Thinking

The framework of computational thinking would not be complete without consideration of the learning theories embedded within the process. CT is meant to be a constructivist process. The idea is to take problems, have students pull them apart, and develop algorithms or methods to solve the problems. The students analyze the results for errors, biases, and appropriate solutions (Wolfram, 2020). To consider a problem, the student contemplates what they know and what they need to know. They work with that knowledge or lack of knowledge, guided by a teacher to scaffold understanding, and work through any misconceptions as needed (Armoni, 2011; Resnick et al., 2009). Ultimately the students come to their own conclusions, “they interpret what they hear in the light of their own knowledge and experience” (Ackermann, 2001). CT uses several constructivist skills to allow students to create their own understanding, work with their own data and content, and generate their own product.

Ackerman (2001) described a main goal of constructivist learning, “learning, especially today is much less about acquiring information or submitting to other people’s ideas or values, than it is about putting one’s own words to the world, or finding one’s own voice, and exchanging our ideas with others”. This is the way that CT is intended to work for the learner. The way to a solution does not have be the same for each student. Students can use methods that are helpful and useful to them. They can collaborate with others, to work with individual strengths and develop a creative or informative solution. Students are not limited to the rigid structures of particular methods, and the goal may be to develop their own method beyond their current understanding (Barr & Stephenson, 2011). The student grows and explores methods with a teacher’s support and guidance. The teacher facilitates the ways the student can develop their own thinking, to grasp new ideas and explore the technology that is available to them (Armoni, 2011).

An even more relevant learning theory associated with CT is constructionism. Seymour Papert, an early proponent of CT, is attributed with the development of constructionism (Ackermann, 2001). Papert was a student of constructivism (Papert, 1980), and advocated for using technology with the metacognitive and knowledge creation elements of constructivism, “Papert’s approach helps us understand how ideas get formed and transformed when expressed through different media, when actualized in particular contexts, when worked out by individual minds” (Ackermann, 2001). He arrived at the idea that computers should power constructivist learning:

> “When a child learns to program, the process of learning is transformed. It becomes more active and self-directed. In particular, the knowledge is acquired for a recognizable personal purpose. The child does something with it. The new knowledge is a source of power and is experienced as such from the moment it begins to form in the child's mind.” (Papert, 1980)

The tools available through technology encourage students to think more about the problems explored. This is the kind of thinking that drives computational thinking, that computers are a tool with which

students can explore metacognition and increase communication in learning. “The intellectual environments offered to children by today's cultures are poor in opportunities to bring their thinking about thinking into the open, to learn to talk about it and to test their ideas by externalizing them" (Papert, 1980). Papert was not interested in the ‘drill and practice’ use for computer-based learning that was prevalent in the early 1980s and is still prevalent today.

> “the idea of the computer as an instrument for drill and practice that appeals to teachers because it resembles traditional teaching methods also appeals to the engineers who design computer systems: Drill and practice applications are predictable, simple to describe, efficient in use of the machine's resources.” (Papert, 1980)

To Papert, computer-based learning should occur through metacognition and construction of tangible and useful artifacts. In relation to constructivism, “Papert’s constructionism, in contrast, focuses more on the art of learning, or ‘learning to learn’, and on the significance of making things in learning” (Ackermann, 2001).

CT is a process that allows students to explore problem solving with a constructivist / constructionist framework. The process is useful in many subject areas, but particularly where problem solving could include technology skills. The technology aspect extends the application of various problem solving methods and gives students the ability to think beyond the limitations of basic skills and knowledge regurgitation.

Computational Thinking and Middle Years Mathematics

Computational thinking is a useful problem solving and thinking process to incorporate into middle years (grades 6-9). Piaget in his stages of cognitive development, describe these students as moving out of the concrete operational stage and into the formal operational stage (Driscoll, 2005). Students in this age range can apply logic to situations, think more abstractly, and synthesize disparate concepts to help them construct their own knowledge (Driscoll, 2005). Socially and mentally, middle years students are defining their identities. They are exploring topics with developing interests, personalities, and attitudes. They are given new opportunities to explore subject areas that they may not have had access to in the early years of school. Their experiences in band, shop, technology, and physical education become more specialized. Students of this age range are more open than older students to trying new subject areas like programming (Kong et al., 2018). It is an interesting time, and a good place to explore tasks and projects that tap into the elements of computational thinking. The following sections reflect upon areas of learning where CT practices are already present in middle years classrooms, and areas which would allow for integration within already existing classroom practice.

CT Applications in Mathematic Instruction: The Math(s) Fix

As much as technology is available to teachers and students in many classrooms, our curricular expectations do not make much room for technology usage. According to Wolfram (2020), “Every maths curriculum around the world starts from the assumption that the student needs to calculate themselves and only then in some cases — if they're lucky — might they progress to using calculating machinery". The focus on computation in our current mathematics instruction leads to a protracted, less dynamic engagement with mathematics (Wolfram, 2020). Papert (1980) raised similar concern that computer use in education is more about “The computer programming the child” rather than “The child programs the computer”.

Wolfram (2020) provided a four-step computational thinking process: define, abstract, compute and interpret the results. To undertake the compute step in CT is to have something else do the calculation rather than the problem solver. This provides more time and capacity for the learner to focus on the other elements of the CT process. Avoiding the drudgery of hand-calculation encourages algorithmic thinking, the exploration of more complex problems, and creativity in problem solving and solution presentation (Wolfram, 2020). In traditional mathematics lessons, students are often provided the problems and their task is to determine the answer. CT shifts the focus from just calculation to problem deconstruction and analysis of solutions, “What was the hardest, most human-centred step in the maths process is now the cheapest and the most mechanized” (Wolfram, 2020). Students need more time in our math classes to ensure they are just as effective at problem building and problem analysis as they are with numeracy and calculation (Wolfram, 2020). "Traditional maths in school has largely drowned out context of the application of the maths from being integral to what's learnt" (Wolfram, 2020).

CT Connections to Real-World Problems

A common question from students within middle years and high school mathematics is “when am I ever going to use this?”. Certain topics within mathematics courses are useful for trades work, further mathematics, science, cooking, and various other topics. However, some of the skills are enigmatic to the learner. Computational thinking allows for focus on real problems while incorporating the abstract nature of certain mathematical concepts, whether that is working with data sets tracking COVID cases or determining how many basketballs would be needed to fill the gymnasium. These types of tasks lead to tangible situations where the search for a solution leads to hands-on activity or the provision of useful tools to showcase and share results. The amount of data and variety of cases that are available to us from all areas of life could easily provide connections across the mathematics curriculum, and into other subject areas. The limitation at this point is not in data and case studies applicable to this age group, the

limitation is the focus on hand-calculation, or even calculator-based analysis (Wolfram, 2020). CT is a way to take problem solving into new realms for students.

Another common concern heard from students relates to frustration with their abilities in mathematics, “I can’t do this!”. Where rote instruction and the focus on calculation perpetuates this kind of thinking in students, the problem solving nature of CT has the potential to break through that kind of thinking. To students who are frustrated, computational thinking provides a process that can open a toolbox beyond hand-calculation. Students can demonstrate knowledge and understanding through their application of tools and their ability to organize or showcase data. Students could show their understanding of parabolas through art and parabola translation using a graphing application on a tablet. Students could show social science research data through an infographic designed online. By expanding the focus of math beyond the limits of calculation, students can build confidence in associated problem building and analysis processes. Calculation is important and has a place within mathematics instruction, but we have tools to better include various modes of thinking that are excluded when the focus is only on calculation.

Programming and Mathematics – The Two Go Together

Programming skills are only a portion of the CT toolkit, but there is still a place for understanding computer programming within mathematics instruction. Grover et al. (2015) propose that there is a strong reciprocal relationship between the development of mathematical skills and computer science skills, “Given the synergies between these domains of thinking and problem solving, perhaps there is also a case for teaching Math through computing and vice versa” (Grover et al., 2015). CT is most effective if students have the skills of programming and coding in their toolkit, “All of today’s students will go on to live a life heavily influenced by computing, and many will work in fields that involve or are influenced by computing” (Barr & Stephenson, 2011).

The provision of programming, computational thinking and computer science education for K-9 students is expanding around the world (European Commission. Joint Research Centre., 2016). It can be difficult to bridge the gap between fun coding activities in elementary school, and the more academic aspects of computer science in high school. It can be difficult for students to make connections between computer programming and other subjects if the computer science instruction is without context. This can lead students to dismiss computer science in high school (Grover et al., 2014). A computer-based instruction that focuses on integrating CT with problem solving in mathematics and other subject areas can provide necessary context. "To remedy the situation, therefore, student perceptions of the discipline of [computer science] must develop early on in their school career and must also move beyond hardware, software, and programming to encompass a more realistic, broader, real-world context" (Grover et al., 2014). To instruct with CT, it is important to include programming, but the reliance can not be programming apart from the rest of the process. The teaching and application of CT comes from the focus on problem solving, real-world applications, and working with interesting topics for the students (European Commission. Joint Research Centre., 2016). Callysto, an online learning tool provided by Cybera, provides a variety of lessons incorporating coding, computational thinking and data analysis across several subject areas (*How It Works – Callysto*, 2021). The tool includes activities incorporating CT and programming to study the bubonic plague, French verb coding, and poetry concepts, to name a few (*Learning Modules – Callysto*, 2021). Learning programming will expand the student's toolkit, and hopefully increase their understanding of the role of computer science in a host of curricular domains.

Probability and Statistics Through Data Science and Computational Thinking

With the ever-increasing power of computers to do the heavy work of data analysis and interpretation, data science can be explored through middle years probability, statistics, digital

citizenship, and social studies. Students can easily incorporate data science to work with real world data, data representation, and integrate large datasets located on or scraped from the web.

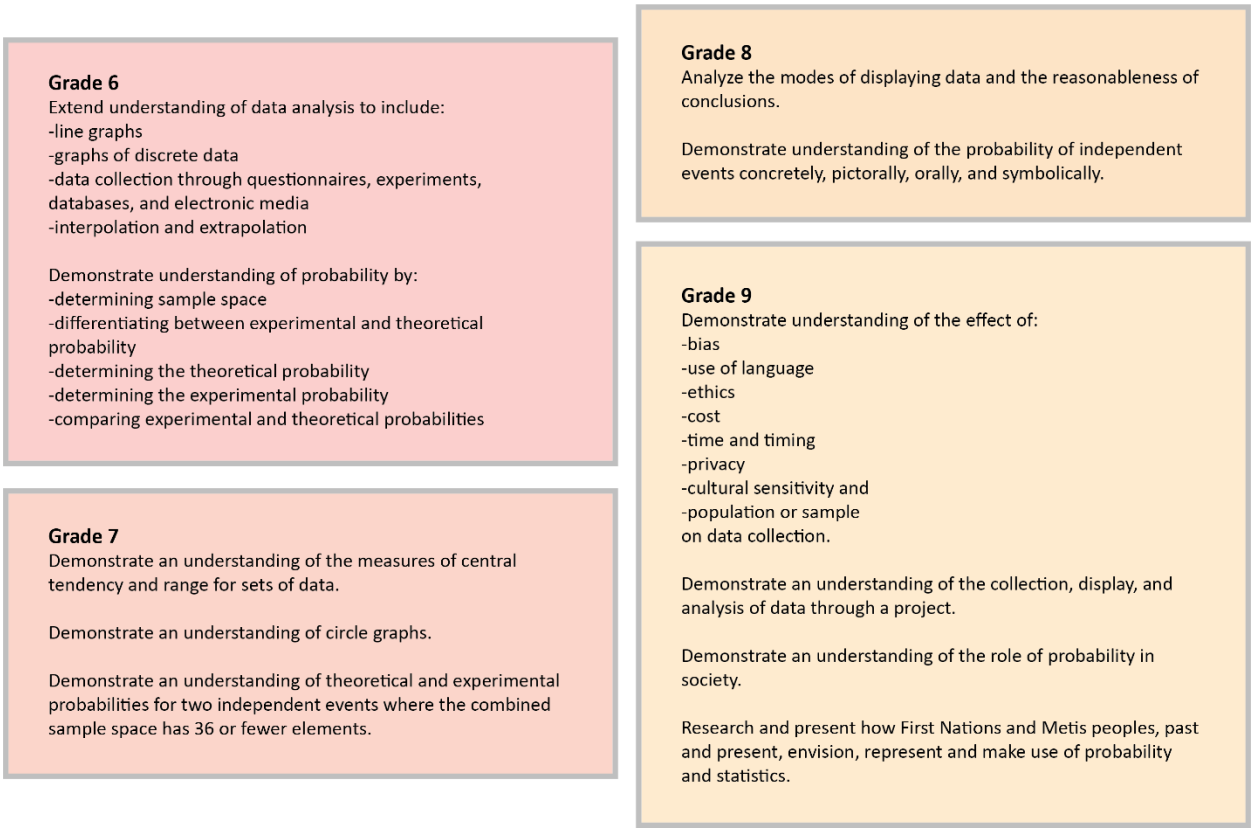

Figure 4: Statistics and Probability Outcomes in Saskatchewan Curriculum (Saskatchewan et al., 2007, 2008; Saskatchewan, Ministry of Education, et al., 2009; Saskatchewan, Saskatchewan Science and Technology, et al., 2009)

The statistics and probability elements of the Saskatchewan curriculum include a surface exploration of the topic of data, see Figure 4. A computer-based analysis would allow for expansive explorations of data sets while maintaining the core competencies as described in the curriculum. The curriculum includes discussion about conclusions, biases, and understanding of the role of probability in society (Saskatchewan et al., 2009). Computer-based data analysis as part of the computational thinking process can deepen those discussions. The ability to explore data confidently, and competently is a necessary skill in our data-flooded world, a world that frequently showcases misinformation. Any ability to overcome misinformation is useful for society. Having students use their own critical thinking, and

tools such as data analysis and visualization to come to their own conclusions about relevant topics is essential for such a vulnerable and connected group of students.

Students need access to the tools and skills related to data analysis. Probability and statistics education is a start, and integrating more computer based analysis and visualization is possible with the tools currently available.

A further application of the probability and statistics curriculum that fits with CT is art and design as part of data visualization (Saskatchewan et al., 2008, 2009). Adding elements of artistry to mathematics is something that students may have little experience with. Teaching data visualization with CT extends how students can represent data, and students have new opportunities to create. They can learn of ways that they could represent their work through graphing or plotting in digital and analog forms. They have the choice then to communicate with data in ways they find suitable: They could choose an analog method like a poster or hand-drawn graphs and charts, but with access and understanding of computer-based data visualization, students could develop interactive digital tools. Though not the only tools available, Jupyter Notebook (Project Jupyter, 2019) and Wolfram Language (Wolfram, 2021) provide programming environments with accessible methods for working with and representing data. With a greater focus on CT processes that allow for data analysis and data visualization, students could be at the forefront for creating community information portals. These young students could explain trends on pandemic information and become leaders in providing reliable accurate information.

Computational Thinking and Game Design

Having a positive attitude about a subject, by both teacher and students, can help students excel. This is further enhanced when students can connect to the content and the purposes of learning. Finding meaning in learning can come in a variety of forms, but if students can see programming and computing as useful, they will likely be more invested (Kong et al., 2018). Many children enjoy games,

find ways to use their devices to pass time, and use their devices to connect with others. These are pastimes and interests that can be explored in education too. Tools like Scratch, and other game design tools, can create a connection between pastime and learning, between games and programming (Resnick et al., 2009). Furthermore, game design connects CT to processes within mathematics, science and associated fields, "One of the main motivations for bringing game design and development into the fold of STEM curriculum planning concerns the need to introduce and familiarize youth to the principles of computation, design thinking and procedural logic, from an earlier age than is currently practised" (Jenson & Droumeva, 2016). Game design incorporates several of the same skills as CT (and mathematics) – planning, problem solving, abstraction, and creative representation.

Game design encourages exploration of the non-computed products of the CT process. For example, many students have had their fair share of time practicing math facts online. An alternative to this would be to have students develop their own tool for practicing math problems. The students would need to decompose the types of questions they would be incorporating within the game. They would have to think about their own learning so that they could incorporate that understanding into the product. They develop a product that would work for them, and maybe for others. This may be time consuming, but the process incorporates more thinking and active learning elements than running through an endless series of questions online.

Game design can take many forms, which provides a variety of opportunities for exploration (Jenson & Droumeva, 2016). The nature of game creation is that game building involves story-building, it can involve a variety of topics, and games do not need to tend towards a particular gender stereotype or form of expression.

Computational Thinking & Project Based Learning

Another strategy that can incorporate both mathematics and computational thinking is project-based Learning (PBL). Project-based learning leads towards the creation of a product, a product that is

personal and shareable (Liu & Hsiao, 2002). As with CT, the PBL process is as important as the content taught through PBL, “Students are engaged in a variety of activities from brainstorming ideas, gathering data, researching information, writing, creating art works, to programming and evaluating” (Liu & Hsiao, 2002). The PBL process is comparable to computational thinking. The processes are constructivist in nature, with students leading the exploration of their learning. Students work towards a final product, but how they get there is up to them, with some likely facilitation. Encouraging students to own their learning is necessary for PBL to be successful, the same applies to CT.

Computational thinking can give middle years students opportunity to see new ways to explore problems with a variety of new tools. CT could allow students to know and explore a new mathematics learning environment, a learning environment where they are undertaking relevant and useful work. A mathematics where students are able to show more than simply their ability to calculate, “Surely we need students to be first-rate problem-solvers, not third-rate human computers” (Wolfram, 2020).

Computational Thinking, Teacher Experience and Learning

Computational Thinking (CT) is a practice that works in an environment of supervisory facilitation (Lye & Koh, 2014). The teacher's role is not primarily to deliver content, but to scaffold concepts and provide guidance to students using tools of all kinds. Teachers can ensure that their environment includes the freedom and constructivist principles to help students develop artifacts through exploration of new and old ideas. The problem-solving nature of CT is well within the scope of teaching through constructivist principles. Educators of middle years and high school students provide this form of instruction regularly. The application of programming within the classroom specifically, is a part of computational thinking that may reside outside of most teachers' skillsets (Caeli & Bundsgaard, 2020; European Commission. Joint Research Centre., 2016; Floyd, 2020; Grover et al., 2015; Kong et al., 2020; Mavroudi & Divitini, 2017). Research by Gülbahar and Kalelioğlu (2017) reported that many teachers who have a computer science background self-reported as insufficient computer science teachers. If a teacher with a computational background self-reported as inadequate, it stands to reason that a grade 7 generalist teacher would likely feel inadequate as well.

For computational thinking to thrive in our classrooms, there needs to be investment into the training and education of all teachers in computational thinking and programming (Caeli & Bundsgaard, 2020; Floyd, 2020; Mavroudi & Divitini, 2017). To ensure that students are empowered, first the teachers need to feel empowered to develop knowledge, skills and confidence through their own coding experiences (Floyd, 2020). As technology continues to evolve, teacher training around computational thinking and programming must be continuous. Knowledge must continually be refreshed as programming language popularity changes, new technologies become available, and the tools become more accessible (Hubwieser et al., 2015).

Teachers must be given opportunities to gain experience with programming, and with the pedagogical content knowledge required for teaching computer science areas and effectively using

computers as tools to further educational goals (Armoni, 2011). Teachers need opportunities to engage with CT in a constructivist environment themselves to ensure that they grasp the best methods for instruction. Armoni (2011) found that deficient pedagogical knowledge around constructivism affected teaching quality, and that students were less likely to engage with higher order thinking when taught with a more traditional process of knowledge transmission. If students are to engage with the higher order thought processes of CT, teaching must not just be about how to code, but how to think through the process.

Though CT is a process that includes more than programming, teachers need to feel confident and comfortable with using technology for teaching. If CT is not well understood by the teacher, and technical skills remain intimidating, they will not want to teach with CT. As nations and educational bodies increasingly integrate CT into classroom instruction, there must be consideration of the additional expectations that are being placed on teachers (European Commission. Joint Research Centre., 2016; Floyd, 2020; Mavroudi & Divitini, 2017; Yadav et al., 2017). "Embedding computational thinking in K–12 teaching and learning requires teacher educators to prepare teachers to support students' understanding of computational thinking concepts and their application to the disciplinary knowledge of each subject area" (Yadav et al., 2017).

Professional Development

Where CT is being implemented, professional development on CT, programming, and computer science is essential (Floyd, 2020). Teachers need time with programming and digital tools. There are several grassroots organizations that provide professional development for coding activities. Organizations like Canada Learning Code, Kids Code Jeunesse, and SaskCode provide teachers and school-based leadership opportunities to explore coding through online platforms, robotics kits, lessons packs, and teacher training (Canada Learning Code, 2018; Kids Code Jeunesse, 2021; SaskCode, n.d.).

While grassroots workshops are an excellent place to start, there are some limitations to their impact (Kong et al., 2020). As schools, divisions and ministries of education continue to roll out the incorporation of CT in classrooms, they will need to ensure that teachers get the support they need. PD must be more than one-off workshops. Teachers need to be active participants, getting hands-on use of the tools available to them on a regular basis. Teachers need to be able to practice with the tools outside of the classroom through PD times. Teachers also need instruction about the pedagogical content knowledge, and technical tools available to them (Kong et al., 2020). Teachers have the tendency to get lost in the technical facets of CT and miss out on the opportunities to break the process down to its component parts and elements.

Where CT skills are integrated into curriculum and practices, the provision of continuous teacher training and the development of a network of experts would ensure the implementation would be more successful. "There is broad consensus among experts and practitioners that the introduction of CT in school curricula at all levels is creating demand for large-scale in-service continuous professional development" (European Commission. Joint Research Centre., 2016). In England, as the new computing curriculum expanded across the country, they trained computer science master teachers to work with groups of teachers in their surrounding communities. Master teachers received 5-10 days of training over the course of six months. They were then responsible to help train pockets of teachers in their regions (European Commission. Joint Research Centre., 2016). This type of training could extend the introductions to the topics made through grassroots workshops.

Teacher Education

The study of computational thinking through pre-service teacher education could provide the necessary instruction to see computational thinking flourish in our classrooms. In locations like Ontario, where CT is now part of the curriculum, the education program at the University of Western Ontario offers CT courses to teacher candidates (Floyd, 2020). Teachers entering the teaching field may have

experience with technology, but many find teaching technical skills required for CT well outside their expertise. As CT is included in curriculum, teachers will need to develop a level of comfort with technology and technology-based learning. Teachers of all interest backgrounds will need to have their beliefs around the relationship between their subject area and technology challenged (Armoni, 2011). It is probably best to do this when they are new to it. More technology is available in the classroom, and teachers should receive a least a cursory look at the potentials of computational thinking and programming during their training. Pre-service teachers should feel comfortable teaching with computational thinking, and without. They should have the opportunity to discuss and experience the use of technology to drive the learning of constructivist principles, and be aware of the limitations (Yadav et al., 2017).

Ensuring that pre-service teachers have experience with CT can take on several forms. Computer lab times could ensure familiarization with CT, especially for teachers who want to specialize in computer science or other CT-centric areas (Armoni, 2011). Lye and Koh (2014) suggest that the optimal learning environment for CT is "a constructionism-based problem-solving learning environment, with information processing, scaffolding and reflection activities" that "could be designed to foster computational practices and computational perspectives" (Lye & Koh, 2014). A hands-on learning environment, with time and resources to practice coding, robotics, 'tinkering' and explicit instruction, would go a long way towards ensuring that teachers have the technical skills and pedagogical principles for incorporating computational thinking. A participatory environment where teachers can work together to explore tasks is one mode with which computational thinking could be explored, "Computational thinking and programming are social, creative practices. They offer a context for making applications of significance for others, communities in which design sharing and collaboration with others are paramount. Computational *thinking* should be reframed as computational *participation*" (Kafai, 2016). Kafai, Lye and Koh explain that this collaborative 'tinkering' workspace is an

optimal situation for learning CT as a student. Providing pre-service teachers with the same participatory environment and artifact building process to teach the foundations of CT would provide a foundation to ensure that constructivist principles around problem-solving are part of the teacher's toolkit.

Ensuring that pre-service, and in-service teachers have experience with computational thinking for any size of region is a large undertaking. However, this training is necessary for computational thinking and programming to expand. Various grassroots organizations have been encouraging the use of programming and various CT skills across grades within K-12. As larger regions, like Ontario, have integrated CT into curricular expectations, the knowledge of teachers has not kept pace (Floyd, 2020), this is reflected in other spots around the world (Caeli & Bundsgaard, 2020). Divisions or departments of education will need to consider providing opportunities for teachers to gain experience with CT to ensure that CT is seen as a current, relevant, and progressive educational concept.

Conclusion

Computational thinking is a constructivist educational process that encourages and expands the ways in which tools and thinking are used for problem solving. The process incorporates computer-based tools when they fit within the process. Though CT remains a broad and non-standardized area of learning, the lack of a specific definition does not take away from the usefulness that CT can bring to any area where problem solving is necessary. As computer-based tools continue to become more accessible and available to our students, the understanding of computer-based learning within schools will be transformed. By applying the principles of CT, mathematics education can expand to focus on a greater diversity of skills. Where data analysis or real-world problems arise across the curriculum, CT can be used to connect computers-based analysis to any subject area. CT is a process that provides agency to the student to explore their own data, to use methods of representation they see applicable, and to work with others to provide artifacts useful to them and others. Computational thinking has potential to engage students with the thinking skills and methods necessary to thrive within our evolving educational environment.

References

Ackermann, E. (2001). *Piaget's Constructivism, Papert's Constructionism: What's the difference?* http://learning.media.mit.edu/content/publications/EA.Piaget&hx0025;20_&hx0025;20Papert.pdf

Aho, A. V. (2012). Computation and Computational Thinking. *The Computer Journal*, *55*(7), 832–835. https://doi.org/10.1093/comjnl/bxs074

Armoni, M. (2011). Looking at Secondary Teacher Preparation Through the Lens of Computer Science. *ACM Trans. Comput. Educ.*, *11*(4). https://doi.org/10.1145/2048931.2048934

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.

Caeli, E. N., & Bundsgaard, J. (2020). Computational thinking in compulsory education: A survey study on initiatives and conceptions. *Educational Technology Research and Development*, *68*(1), 551–573. https://doi.org/10.1007/s11423-019-09694-z

Canada Learning Code. (2018). *Teaching Code*. Canada Learning Code; Canada Learning Code. https://www.canadalearningcode.ca/teaching-code/

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). *Computational Thinking—A Guide for Teachers*. Computing At School.

Dagienė, V., Sentance, S., & Stupurienė, G. (2017). Developing a Two-Dimensional Categorization System for Educational Tasks in Informatics. *Informatica*, *28*(1), 23–44. https://doi.org/10.15388/Informatica.2017.119

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

Driscoll, M. P. (2005). *Psychology of learning for instruction* (3. ed). Pearson Allyn and Bacon.

Einhorn, S. (2012). MicroWorlds, Computational Thinking and 21st Century Learning. *LCSI White Paper*.

European Commission. Joint Research Centre. (2016). *Developing computational thinking in compulsory education: Implications for policy and practice.* Publications Office. https://data.europa.eu/doi/10.2791/792158

Floyd, L. A. (2020). The Integration of Coding in Teacher Education Programs: Course Experiences and the Teaching and Learning of Mathematics. *Proceedings of the 2020 ACM Conference on International Computing Education Research*, 350–351. https://doi.org/10.1145/3372782.3407105

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, *25*(2), 199–237. https://doi.org/10.1080/08993408.2015.1033142

Grover, S., Pea, R., & Cooper, S. (2014). Remedying misperceptions of computer science among middle school students. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education - SIGCSE '14*, 343–348. https://doi.org/10.1145/2538862.2538934

Gülbahar, Y., & Kalelioğlu, F. (2017). Competencies of High School Teachers and Training Needs for Computer Science Education. *Proceedings of the 6th Computer Science Education Research Conference on ZZZ - CSERC '17*, 26–31. https://doi.org/10.1145/3162087.3162092

How It Works – Callysto. (2021). Callysto; Callysto. https://www.callysto.ca/how-it-works/

Hubwieser, P., Giannakos, M. N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., Pal, Y., Jackova, J., & Jasute, E. (2015). A Global Snapshot of Computer Science Education in K-12 Schools. *Proceedings of the 2015 ITiCSE on Working Group Reports - ITICSE-WGR '15*, 65–83. https://doi.org/10.1145/2858796.2858799

Jenson, J., & Droumeva, M. (2016). Exploring Media Literacy and Computational Thinking: A Game Maker Curriculum Study. *Electronic Journal of E-Learning*, *14*(2), 111–121.

Kafai, Y. B. (2016). From computational thinking to computational participation in K--12 education. *Communications of the ACM*, *59*(8), 26–27. https://doi.org/10.1145/2955114

Kids Code Jeunesse. (2021). *Code in the Classroom | KidsCodeJeunesse.org*. Kids Code Jeunesse; Kids Code Jeunesse. https://kidscodejeunesse.org/code-in-the-classroom

Kong, S.-C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education*, *127*, 178–189. https://doi.org/10.1016/j.compedu.2018.08.026

Kong, S.-C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education*, *151*, 103872. https://doi.org/10.1016/j.compedu.2020.103872

Learning Modules – Callysto. (2021). Callysto; Callysto. https://www.callysto.ca/learning-modules

Liu, M., & Hsiao, Y.-P. (2002). Middle School Students as Multimedia Designers: A Project-Based Learning Approach. *Journal of Interactive Learning Research*, *13*(4), 311–337.

Lye, S. Y., & Koh, J. H. L. (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

Mavroudi, A., & Divitini, M. (2017). Enabling factors and self-efficacy: The case of Norwegian computer science teachers. *Proceedings of the 6th Computer Science Education Research Conference on ZZZ - CSERC '17*, 32–37. https://doi.org/10.1145/3162087.3162093

Palts, T., & Pedaste, M. (2020). A Model for Developing Computational Thinking Skills. *Informatics in Education*, *19*(1), 113–128. https://doi.org/10.15388/infedu.2020.06

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

Project Jupyter. (2019). *Jupyter*. Jupyter.Org. https://jupyter.org/

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Saskatchewan, Department of Learning, & Curriculum and E-Learning Branch. (2007). *Grade 7 mathematics: Curriculum.* Saskatchewan Learning.

Saskatchewan, Ministry of Education, Curriculum and E-Learning, & Science and Technology Unit. (2008). *Mathematics 8.* Mininstry of Education, Science and Technology Unit.

Saskatchewan, Ministry of Education, Saskatchewan, & Saskatchewan Science and Technology. (2009). *Mathematics 6.* Saskatchewan, Ministry of Education.

Saskatchewan, Saskatchewan Science and Technology, Saskatchewan, & Ministry of Education. (2009). *Mathematics 9.* Saskatchewan, Ministry of Education.

SaskCode. (n.d.). *SaskCode*. SaskCode. Retrieved January 1, 2021, from https://www.saskcode.ca/pd-events

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M., & Stanzione, D. (2016). Progress in computational thinking, and expanding the HPC community. *Communications of the ACM*, *59*(7), 10–11. https://doi.org/10.1145/2933410

Wolfram. (2021). *Wolfram Language: Programming with Built-in Computational Intelligence*. Www.Wolfram.Com; Wolfram. https://www.wolfram.com/language/

Wolfram, C. (2020). *The math(s) fix: An education blueprint for the AI age*.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, *60*(4), 55–62. https://doi.org/10.1145/2994591