Assessing Computational Thinking Across the Curriculum

Julie Mueller, Danielle Beckett, Eden Hennessey, and Hasan Shodiev

Abstract Computational thinking (CT) refers to a set of processes through which people arrive at solutions to problems using principles based in computer science. A CT approach to problem-solving is increasingly valuable in education and workplace settings as the economy grows more dependent on digital literacy. Given the importance of CT, it is essential to assess these skills. However, a reliable assessment tool is absent from the current literature. This chapter, therefore, defines CT across the Ontario (Canada) Elementary School curriculum in elementary classrooms and addresses the need for effective instructional strategies and assessment of CT-related problem-solving abilities. Finally, we establish where CT concepts and skills already exist or are missing from the curriculum and suggest a workable tool to assess CT based on existing literature.

Keywords Assessment of problem-solving • Computational thinking (CT) • Curriculum expectations • Elementary education

Introduction

A growing digital economy and the need for an ever increasing percentage of the population to work with twenty-first century skills (e.g., problem-solving, creating, collaborating, communicating, critical thinking) demand that these skills be addressed and supported starting from an early age and that they remain a focus as students develop (Dede, 2010). Learning theory for a digital age emphasizes authenticity, audience, and authorship—children are creating, sharing, and learning with purpose (Bellanca & Brandt, 2010). The Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) (2011)

J. Mueller (✉) • E. Hennessey • H. Shodiev
Wilfrid Laurier University, 75 University Ave W, Waterloo, ON N2L 3C5, Canada
e-mail: jmueller@wlu.ca; henn8280@mylaurier.ca; hshodiev@wlu.ca

D. Beckett
Brock University, 1812 Sir Isaac Brock Way, St. Catharines, ON L2S 3A1, Canada
e-mail: daniellebeckett@gmail.com

P.J. Rich, C.B. Hodges (eds.), *Emerging Research, Practice, and Policy on Computational Thinking*, Educational Communications and Technology: Issues and Innovations, DOI 10.1007/978-3-319-52691-1_16

suggest that the following dispositions or attitudes accompany these skills: “confidence in dealing with complexity, persistence in working with difficult problems, tolerance for ambiguity, the ability to deal with open-ended problems, and the ability to communicate and work with others to achieve a common goal or solution” (p. 7). Although these skills and dispositions do not require technology, they are supported by and encouraged in a digital environment. Computational thinking and problem-solving are related to higher-order thinking skills that are recognized as fundamental to success in a digital age.

Computational thinking can be considered a specific type of problem-solving (i.e., approaching a problem with a particular mind-set utilizing computer technology). Computer programming involves the identification of a problem and the creation of a solution using a language and logic that directs a computer to perform actions leading to that solution. Computational thinking (CT) takes computer science outside of the computer lab and makes it accessible to everyone, rather than computer programming, often seen as a narrow and “tedious, specialized activity, accessible only to those with advanced technical training” (http://scratch.edu, 2016).

Considering and using computational thinking across disciplines to solve problems places computer programming within the reach of students at any age. A change in emphasis in learning, from knowledge acquisition to higher-order knowledge construction, makes it important for teachers to transform practice to approach computational thinking for students in all disciplines and not solely in computer science. Introducing creative and critical thinking is not a brand new endeavor for teachers (Griffin, 2014), but it can be more deliberate, specifically recognizing when a computer can help us to gather, analyze and manipulate data, create simulations, and persist with complex and difficult problems (Barr & Stephenson, 2011; CSTA & ISTE, 2011; Voskoglou & Buckley, 2012; Wing, 2006).

What is missing from the literature, and most learning frameworks, however, is a valid and reliable assessment of computational thinking skills. Some research recommends using multiple assessments in a “systems of assessments” approach to assessing computational thinking (Grover, 2015). However, employing many assessment tools can be costly and onerous. We, therefore, propose a more comprehensive assessment of computational thinking skills to provide an understanding of how such skills may be applied across disciplines.

Computational thinking is not, and should not be, an additional area of curriculum content, but rather an integrated component of already existing curricula. Teachers may not be prepared to include more content in what many describe as an “overcrowded” curriculum. One school principal, Brian Nichols, summarized an online “#edchat” (i.e., a weekly Twitter discussion of educators; 2010) on “how to manage standards and an overloaded curriculum,” using three key themes: preparedness, essentiality, and integration. He suggests that in “covering the curriculum,” teachers need to choose content and skills that prepare students for a workforce dependent on “the ability to create new ideas, synthesize information, and problem solve with people all over the globe.” Further, teachers must identify what is “essential” and integrate such skill sets across disciplines rather than addressing problem-solving strategies in isolation.

This chapter identifies a working definition of computational thinking across the curriculum in Ontario's elementary classrooms and addresses the need for effective instructional strategies and assessment of problem-solving abilities. Further, we analyze where CT concepts and skills already exist or are missing from the Ontario Elementary School curriculum (Ministry of Education, Ontario, Canada). Finally, we suggest a workable CT assessment tool based on existing literature and current findings (Brennan & Resnick, 2012; Hesse, Care, Buder, Sassenberg, & Griffin, 2015; Voogt, Fisser, Good, Mishra, & Yadav, 2015; Wilson, Scalise, & Gochyyev, 2015).

What and Where Is Computational Thinking?

A Working Definition

Before a construct can be discussed, debated, and analyzed, an agreed-upon definition is generally a starting point. Wing's (2006) definition of computational thinking is considered to be that starting point although many researchers and educators have reviewed the definition, massaged its components, and set it in context since (e.g., Barr & Stephenson, 2011; National Research Council, 2010; Shelby & Woollard, 2013). In a recently published examination of computational thinking in compulsory education, Voogt, Fisser, Good, Mishra, and Yadav (2016) spoke to the need for a definition of computational thinking that includes "peripheral skills" important to CT but not "necessary and sufficient" if CT is to be implemented in the practice of teachers across disciplines.

Researchers have identified seven core *concepts* that are useful in programming, including sequences, loops, parallelism, events, conditionals, operators, and data. Computational *practices* consist of eight terms and refer to how one is learning: experimenting and iterating, testing and debugging, reusing and remixing, and abstracting and modularizing. Computational *perspectives* capture how programmers' perspectives are impacted during CT in three ways: expressing, connecting, and questioning.

The purpose of this chapter is not to review the historical development of the definition of the term, but rather to set a working definition in context for assessing CT across the curriculum in an elementary school setting using coding as a tool to teach this approach to problem-solving. The challenge is to identify where and when the concepts, practices, and perspectives that define computational thinking are, can, and should be introduced to learners.

Given the importance of computational thinking in the future, it is of related interest where and how much the present elementary curriculum addresses CT and associated processes. Indeed, an understanding of how the existing curriculum already addresses CT can (1) establish a starting place for educators who wish to expand the curriculum to incorporate or expand CT resources and (2) provide educators with evidence of where they already address CT. In effect, we suggest that adapting current teaching materials should not be intimidating. This could increase openness to dialogue around where CT may be "hiding in plain sight" within the current elementary school curriculum.

As an example, we conducted a systematic content analysis of the Ontario Elementary School curriculum (grades 1 through 8) for 38 terms (see Table 1) associated with computational thinking (Brennan & Resnick, 2012; Grover & Pea, 2013; Scratch Ed, 2016; Yadav et al., 2011). Content analysis refers to a research technique for making inferences from data to their context (Krippendorff, 2012). The goal of a content analysis is to systematically review and extract text into meaningful categories, which can then be used to draw conclusions.

Our primary research questions in this content analysis included:

1. *Frequenciesacross subject areas*: How often do the specific phrases "computational thinking" and "problem-solving" appear in the Ontario Elementary School curricula and across which subject areas? How often do CT-related terms (and their iterations) appear in the Ontario Elementary School curricula and across which subject areas?
2. *Grade level*: In which grade levels are students introduced to CT-related terms (and their iterations) across the Ontario Elementary School curricula?
3. *Context/location*: In which sections of the Ontario Elementary School curricula do CT-related terms (and their iterations) appear?

Method

Ontario curriculum documents are accessible to the public in multiple formats. We specifically analyzed the text files versus the print or PDF files given the large volume of data. In total, the Ontario curriculum documents total 1496 pages. Subject areas include kindergarten, mathematics, arts, sciences, language, health and physical education, social studies, French as a second language, and Native language studies (average page count was 187 pages). The Ministry of Education mandates that each year a number of subject areas enter the review process, so they remain relevant and age appropriate. Thus, educators, parents, and students at least once between 2005 and 2015 have reviewed subjects comprehensively.

In the current study in particular, we reviewed kindergarten, mathematics, arts, science and technology, health and physical education, social studies, French as a second language, and Native language studies for frequencies of terms. Then, we narrowed our search to a subset of terms and disciplines (i.e., mathematics, science and technology, language, and arts), focusing on areas relevant to our future applied research in classrooms. We reviewed each subject area for frequency of terms as well as context and location (i.e., curriculum expectations, front matter). Two trained researchers (i.e., graduate students in social psychology and business/computer science) conducted analyses, and any discrepancies were resolved through discussion. As well, we conducted an in-depth content analysis of the context,

Table 1 Computational thinking key terms searched in content analysis

Key term	Derivatives
1. Abstract	Abstracting[a]; abstraction
2. Algorithm	Algorithmic
3. Analyse	Analysis; analyze; analyzes; analyzed
4. Apply	Application; applied; applying
5. Automate	Automated; automatically; automation
6. Code	Coder; coding; coded; codes
7. Collection	
8. Computational thinking	Computational
9. Compute	Computed; computes; computer; computing
10. Conditionals[a]	
11. Connecting	
12. Data	Data analysis; data collection; data representation
13. Debugging[a]	
14. Decompose	Decomposed; decomposing; decomposition
15. Events[a]	
16. Experimenting[a]	
17. Expressing[a]	
18. Generalize	Generalization; generalized; generalizing
19. Identification	
20. Iterating[a]	
21. Logic	Logical
22. Loops[a]	
23. Management	
24. Model	Modeling; modeling
25. Modularizing[a]	
26. Operators	
27. Parallel	Parallelization
28. Problem	
29. Problem-solve	Problem-solves; problem-solvers; problem-solving
30. Questioning[a]	
31. Recursive	
32. Remixing[a]	
33. Representation	
34. Reusing[a]	
35. Sequence[a]	
36. Simulate	Simulated; simulation
37. Technology	Technological
38. Testing[a]	

[a]Terms from Brennan and Resnick (2012) used in specific search.

grade, and location of the terms within the curriculum document (i.e., curriculum expectations or front matter) using only the concepts, processes, and perspectives identified by Brennan and Resnick (2012). Preliminary results of the more specific content analysis of the disciplines are presented here (See Table 2). A more detailed examination of the results within context will be included in a forthcoming article by Hennessey, Mueller, Beckett, and Fisher (Unpublished manuscript).

To first determine frequencies of our terms across the various curricula, we wrote a script using Python (the programming language) to search for each word using regular expression pattern matching (see Goyvaerts, 2016). The script was designed to find all terms associated with CT processes. Included in this larger search were the focal CT definition terms delineated in Brennan and Resnick (2012). The script was designed to match all variations of each word stem in our search list [e.g., we searched for all variations of "code" such as "coding," "coded," "coder," and "codes" with the pattern cod(e|ing|ed|er|es)] while excluding words that were nested within other words which were not of interest (e.g., variations of "sequence" were found; however, variations of "consequence" were excluded). The most up-to-date version of each curriculum was searched with the script. Wherever possible, the plaintext version of the curriculum was searched. However, not all of the most up-to-date curricula have a plaintext version made publicly available. In these cases, the PDF version was first converted to a text file in order to be searched. This conversion process is inherently imperfect, and as such, the final frequency count for these curricula is expected to be a slight underestimation, as the converted text may have been broken up in the conversion process such that the regular expression would no longer match.

Results and Discussion

Frequencies of CT-related terms appear in Table 1. Our analyses showed that while the exact phrase "computational thinking" does not appear in the Ontario Elementary School curricula, the term "computational" alone appears sparsely and only in the mathematics curriculum (15 instances), mostly in a title describing "computational strategies." Iterations of the term "compute" appear more frequently in the mathematics curriculum (30 instances), but are less frequently cited in arts (10 instances), language (11 instances), or the science and technology (3 instances) curricula. The phrase "problem-solve" and its iterations appear with more frequency than "computational thinking" in the mathematics (459 instances) and science and technology (134 instances) curricula; however, "problem-solving" is sparsely mentioned in the arts (76 instances) and language (38 instances) curricula.

Although CT itself is not found explicitly in the current elementary curriculum, there are related terms present across disciplines. Initial frequency analyses indicated that terms associated with CT mostly appeared in the mathematics (1,259 instances) and arts (935 instances) curricula. These terms were also fairly commonly used in science and technology (886 instances). Terms associated with CT appeared somewhat often in the language (522 instances), kindergarten (447 instances), and health and physical education (276 instances) curricula and less frequently in the Native languages (53 instances) document.

A more specific analysis of the mathematics, science and technology, arts, and language documents using the computational *concepts*, *practices*, and *perspectives* from Brennan and Resnick's (2012) definition suggests that the frequencies of terms differ across disciplines and grades and that *concepts* are addressed more often than *practices* or *perspectives*. See Table 2 for frequencies of each term across different subject areas.

All four of the disciplines we examined include the terms "data" and "events." The mathematics curriculum, not surprisingly, uses the term "data" much more frequently than the other three disciplines. Both arts and language, however, also include the term "sequences," while mathematics and science, interestingly, do not. The only other specific CT *concept* that was found in the documents was "operators" (3 instances) in the science and technology curriculum. Any specific CT *practices* were referred to only in the science and technology curriculum—"testing" (19 instances) and "reusing" (3 instances). Only 1 instance of "abstracting" was found in the arts. Frequencies of terms defined as CT *perspectives* (i.e., "expressing," "connecting," and "questioning") were spread more evenly across the documents—

Table 2 Frequency of CT concepts, practices, and perspectives in mathematics, science and technology, language, and arts curricula

Subject area	Mathematics	Science and technology	Language	Arts
Key term concepts				
Data	253	28	5	3
Events	21	12	33	36
Operators	0	3	0	0
Sequences	0	0	3	8
Loops	0	0	0	0
Parallelism	0	0	0	0
Conditionals	0	0	0	0
Practices				
Testing	0	19	0	0
Reusing	0	3	0	0
Abstracting	0	0	0	1
Incremental	0	0	0	0
Iterative	0	0	0	0
Debugging	0	0	0	0
Remixing	0	0	0	0
Modularizing	0	0	0	0
Perspectives				
Connecting	23	2	17	7
Expressing	2	1	4	2
Questioning	1	1	9	4

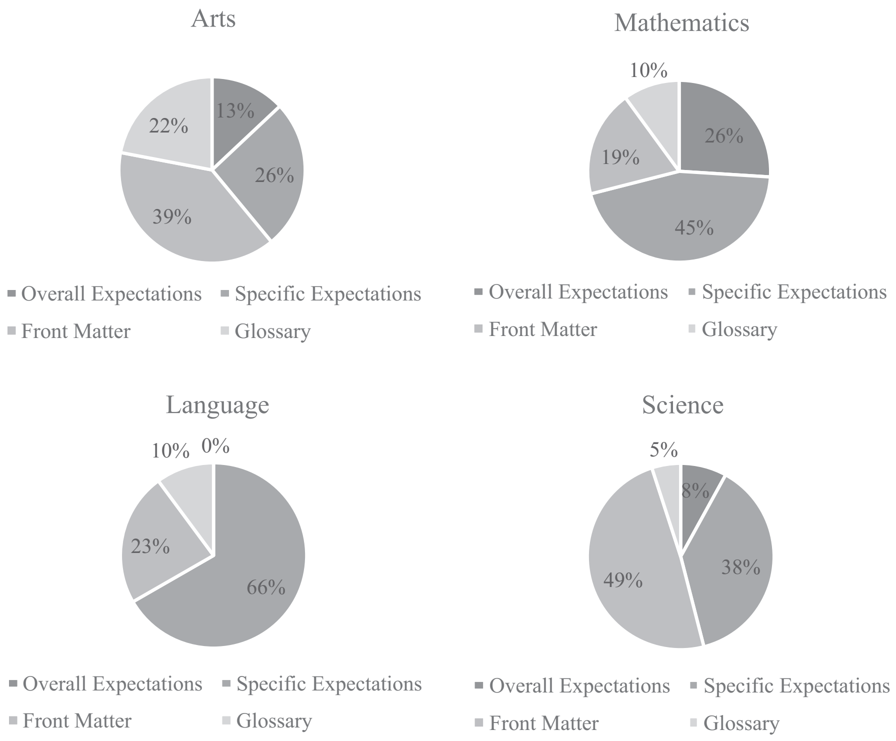

Fig. 1 Percentage of CT terms by location in curriculum documents

all had at least 1 instance of each term, but not in large numbers. The perspective of "connecting" was used most frequently in mathematics, followed by language, while language and arts included "questioning" more often than mathematics or science, each with just a single instance.

This initial analysis of instances of computational thinking terminology within the curriculum across disciplines suggests that terms related to the concepts of computational thinking can be found within the current curriculum, but specific terms unique to computer programming are not. It appears that a questioning and connecting perspective is a part of the current content in the elementary curriculum in the language and arts disciplines, suggesting that students may be learning the *concepts* and *perspectives* that form a foundation for computational thinking. What may be missing are the actual *practices* involved in computational thinking.

Examining "where" instances of CT-related terms are found within the curriculum may shed additional light on the meaning of the analysis. See Fig. 1 for a breakdown of percentages in location within the curriculum document.

The largest percentage of CT-related terms in the language and mathematics curricula is found in specific expectations—the "what" or content that is to be taught—while analysis of both the science and arts curricula indicates that CT-related terms

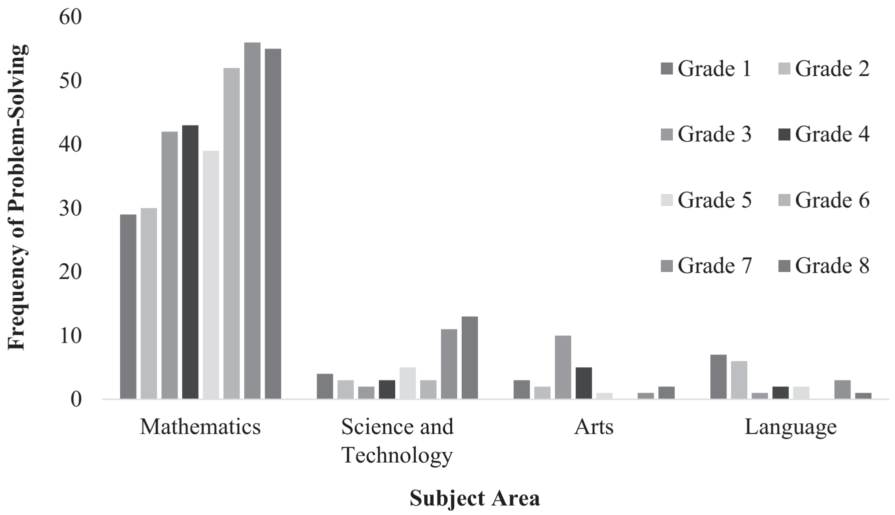

Fig. 2 Frequency of problem-solving in subject areas across grade levels

are most frequently found in the front matter of those documents, the "how" the subject is to be taught, the program planning, assessment and evaluation, and general overview of the discipline.

Aside from the frequency of computational thinking-related terms across disciplines, the developmental sequence across grades is also of interest. "Computational thinking" specifically is not a term used in any of the curriculum documents at any grade, but "solving problems" and "problem-solving" are terms used in increasing quantity within the documents. See Fig. 2 for the progression across grades in each of the four discipline areas examined in more detail.

Politicians, industry leaders (e.g., Google; Wing, 2014; Code.org), parents, and the computer science community are encouraging educators, beginning at the elementary level, to "transform" their teaching practices so that CT is added to current curriculum (Barr & Stephenson, 2011; Repenning, Webb, & Ioannidou, 2010). These types of sweeping changes and calls for transformation in educational practice and content rarely see immediate or substantial change. Identifying what currently "works" and sharing with educators where this way of thinking can be found in their curriculum and what these concepts, practices, and perspectives look like across grades and disciplines serves as a first step in making the necessary adjustments to twenty-first century education. Jun, Han, Kim, and Lee (2014) examined the computational literacy of Korean elementary students using information and communications technology (ICT) literacy as a broader issue, suggesting that the curriculum needs to be revised to include computational problem-solving skills and that teaching methods need to be more accessible and effective for teachers. Preliminary review of the Ontario Ministry of Education Curriculum for elementary students indicates that specific skills and practices used in computer programming may need to be included in revised and updated curriculum, but CT perspectives and ways of questioning may already be present in the curriculum—hiding in plain sight. A key question then is how educators teach these skills, practices, and perspectives to develop computational thinking across the disciplines and grades and how those concepts and skills are measured and assessed.

Teaching Computational Thinking and Problem-Solving Through Coding Across Disciplines

Developments and advancement in programming languages and digital technology have made coding and computer programming more accessible and user friendly, and Dr. Wing's (2006) vision possible. More recently, a debate (see Charlton & Luckin, 2012; Barr & Stephenson, 2011; Naughton, 2012) has emerged suggesting that not everyone can, or should, be a computer scientist, with arguments analogous to those suggesting that you do not need to be a mechanic to drive a car. However, twenty-first century global problems—social, economic, and environmental—demand that our educational institutions develop citizens who are able to approach these problems in creative and innovative ways, refining problems, developing solutions, and evaluating outcomes virtually and in real life (Barr & Stephenson, 2011). The sheer volume of information available and the data involved in solving these problems require computer-supported approaches .

Computational thinking enables the scaling of problem-solving. According to Constable (2005),

> "…Computers change the scale at which resources can be examined, and they already provide sufficient discriminatory powers that scale and speed compensate for their currently limited intelligence as they draw conclusions, make predictions, and participate in discoveries…The challenge for society is to assimilate digital knowledge and to improve the human condition by its application." (p. 1)

Coding has been introduced and perhaps even "hailed" as a panacea to ensure that learners are indeed introduced to, and develop, the ability to solve complex twenty-first century problems using computer programming. US President Obama's 2017 budget, in fact, includes four billion dollars to support computer science in schools, identifying computer science as a "basic skill" in his Computer Science for All initiative (Shear, 2016).

Computational thinking can be made accessible to students and teachers through concrete approaches to computer coding. Utilizing the potential and benefits of computer coding to develop skills and strategies across disciplines and across developmental levels will begin to provide a foundation for the development of computational thinking for everyone. Activities to learn and improve specific skills can be used across grade levels from kindergarten to grade 12, e.g., abstraction, algorithms and procedures, automation, simulation, and parallelization (CSTA & ISTE, 2011). In order to measure the effectiveness of the activities and the impact on problem-solving beyond those activities, tools measuring these twenty-first century skills must be developed, validated, and utilized. Problem-solving can be measured using complex, authentic examples in context or using a confined approach with one or two solution problems within limitations (Voskoglou & Buckley, 2012; Voskoglou & Perdikaris, 1993). Further evidence can then be collected through the use of verbal protocol, interviews, and naturalistic observation, and one can use computers to track strategies and approaches to problems within Web or applications.

Resources have been developed and utilized to support teachers in the kindergarten to grade 12 environment using computer coding to represent problems and collect data (e.g., CSTA & ISTE, 2011). Advances in computer technology have created applications and programs that allow the user to build and create without knowing a complex computer language. One such application was developed at MIT and is available to users online: Scratch (http://scratch.mit.edu). The platform was designed for students from ages 8 to 16 but is used by people of all ages across learning contexts and disciplines. Using "drag, drop, and click" blocks of code, users can build projects to animate, simulate, tell stories, and make music. Utilizing a software application such as Scratch, measuring problem-solving ability before and after its use across curriculum areas (disciplines) and age levels, will allow us to assess the impact of instruction related to computational thinking on problem-solving ability and introduce CT to teachers and students (see Koehler & Ackaolgu, 2014).

The key question is whether learning to code is an effective method for developing problem-solving that transfers across disciplines and contexts. An examination of the curriculum and how CT is addressed, or not, at the elementary level serves as a starting point for measuring computational thinking using a systematic approach.

Measuring Computational Thinking and Problem-Solving

Even when using the most effective teaching methods, teachers cannot assume that learning occurs. It is well documented that students develop and learn at different rates (Angelo & Cross, 1993; Cashin, 1990; Drake, Reid, & Kolohan, 2014; Sternberg, 1986, 2009) and teaching quality varies from classroom to classroom; therefore, teachers should not assume all students have grasped what has been taught (Western and Northern Protocol for Collaboration in Education, [WNPC], 2006). As a result, classroom assessment and evaluation are essential to measure what students have learned. To date, extensive research exists on traditional classroom assessment strategies that promote effective instruction and student learning (Andrade, 2009; Brookhart, 2009; Black, Harrison, Lee, Marshall, & Wiliam, 2003; Black & Wiliam, 2009; Dann, 2014; Earl, 2003; Hattie, 2012).

What's lacking presently in classroom assessment research is the twenty-first century context of classroom assessment. Specifically, with the influx of new skills deemed necessary for the twenty-first century (e.g., computational thinking skills), teachers are at a loss of how to teach and assess such skills (Griffin & Care, 2015; DiCerbo, 2014). In fact, the Association for Computing Machinery (ACM) and the Computer Science Teacher Association (CSTA) stress the major factor that limits computational thinking into schools is the lack of assessments available to teachers.

Computational thinking has a wide application beyond computing itself. It is the process of recognizing aspects of computation in the world and applying tools and techniques from computing to understand and reason about natural, social, and artificial systems and processes (Csizmadia et al., 2015). It allows students to tackle problems, to break them down into solvable chunks, and to devise algorithms to solve them. Therefore, computational thinking concentrates on students performing a thought process, not on the production of artifacts or evidence. This in itself can be problematic for assessment because it is difficult to measure actual thought processes.

Although we advocate for a systems of assessment approach whereby "assessment as, of, and for learning" are used purposely, we recognize that "assessment as learning" plays a large role in effective "assessment of" students' computational thinking. That is, in order for teachers to truly know what and how students are thinking, students are required to demonstrate their thought processes in some way. One of most effective methods for assessing student thinking is the think-aloud method (Ahonen & KanKaanranta, 2015). Two methods are available for collecting student think-aloud data: concurrent and retrospective think-alouds (Ericsson & Simon, 1993). During a concurrent think-aloud, students think aloud as they complete a task with the intent of unveiling the cognitive processes they engage in and the information they attend to while they are solving a problem (Leighton & Gierl, 2007). In contrast, retrospective think-alouds are conducted after students have solved the problem, providing students an opportunity to describe any metacognitive processes in which they engaged while solving the problem (Ericsson & Simon, 1993). Although both methods can provide rich accounts of students' computational thinking, concurrent think-alouds collect data from short-term memory which is preferable when assessing computational thinking skills because they are not tainted by perception, providing the most accurate representation of knowledge and ability. And yet, this method can be costly in terms of time and scale. It is unreasonable to expect teachers to be able to use it writ large. Recently, however, easier accessibility to new and older technologies in schools, such as audio and video recordings and screencasting with student narration, mitigates some of the caveats of the think-aloud method.

Metacognition is often tied with think-alouds as it requires students to think about their own thinking (Flavell, 1979). Think-alouds and metacognitive processes are at the crux of deep, purposefully "assessment as learning." Computational thinking is a systematic thinking process, and for this reason the think-aloud is a valuable method for properly understanding how computational thinking happens at the cognitive level. That is, this method provides teachers with rich student data about how the student is thinking through a problem, consequently allowing the teacher to provide deliberate, personalized instruction to further student learning. However, one of the drawbacks of this method is that younger children are not always aware of their own cognition, and therefore thinking aloud could distract them from the task. This brings to light that metacognition is not an innate ability; it must be taught. Volante and Beckett (2011) suggested a "lockstep process" to teaching AaL; that is, teachers provide guidance to students when cultivating evaluative knowledge and expertise and reflecting on what they have learned. It is beneficial for teachers to provide students with a pool of appropriate strategies to bring their own performance closer to the desired goal (Sadler, 1989). This argument is strongly aligned with Earl's (2003) conceptual framework of the three purposes of assessment in that she believes teachers should establish an environment in which AaL is central to student learning and that all other assessments are rooted in such practices.

The bidirectional transmission between teacher assessment (i.e., AaL and AfL) and student self-assessment (i.e., AaL) is a vital factor for optimal assessment of computational thinking. What remains to be created in the area of computational thinking though is an assessment tool that will allow teachers to evaluate CT and scaffold additional teaching and learning. Before an assessment tool can be developed, a clear description of what is being assessed is required.

We provide a set of teacher verbal protocols to aid in the evaluation of students' computational thinking processes. By asking these types of questions, teachers will be able to gain a deep understanding of students' computational thinking ability and how it relates to problem-solving outside of coding. Specifically, the questions are broken down by processes so that teachers can pinpoint which are areas of strength and areas of needed improvement.

The questions are categorized based on skills and processes that are inherent to computational thinking and as such would be useful skills to transfer across disciplines to solve problems. For instance, in order to evaluate students' algorithmic thinking across disciplines, teachers can ask "Can the student create a set of steps to solve a problem" and "Can the student solve similar problems with the same set of steps or principles?" See Table 3 for a possible list of questions. By using this set of questions, teachers may gain a more comprehensive understanding of students' ability to transfer computational skills and processes into other disciplines and guide further instruction.

This set of questions can also be used as a communication tool between teachers and students. Evaluation as communication can promote student learning and development of computational thinking and problem-solving skills. Specifically, a discussion (with these questions as a guide) can confirm with students the quality of their performance and provide insight on how they can improve and further develop their computational thinking skills. The questioning in and of itself encourages and scaffolds metacognition and computational thinking. With that said, our intention is not to reduce or oversimply the evaluation of computational thinking only to these sets of skills. We recognize this as a starting place and a means to support discussion and provoke thought around how to assess computational thinking skills. Ultimately, computational thinking is a process and therefore should not be evaluated as an end product. It is an ongoing learning progression through grade levels and across subject areas to eventually produce effective and productive twenty-first century thinkers. Future research will examine the "think-alouds" of children as they participate in tasks intended to develop computational thinking to further inform the types of questions to be used in assessment of CT and provide criteria and examples of development across grades and disciplines.

Table 3 Assessment tool for concepts, processes, and perspectives in problem-solving and computational thinking

Skill (source)	Questions
Algorithmic thinking	Can the student create a set of steps to solve a problem? Can the student solve similar problems with the same set of steps or principles?
Decomposition	Can the student break down the problem into smaller, more manageable parts?
Generalization/inferencing	Can the student transfer prior knowledge and skills? Can the student identify patterns, similarities, and connections between prior and current problems? Can the student make inferences?
Abstraction	Can the student evaluate what is valuable information and what is not? Can the student remove unnecessary information? Can the student add or remove details to clarify a problem?
Evaluation	Can the student evaluate if the solution is a good one?
Incremental/iterative thinking	Can the student identify a concept for the project? Can the student develop a design plan? Can the student implement the design plan? Is the student comfortable adapting the plan in response to new or different information?
Testing and debugging	Can the student develop strategies for dealing with problems? Can the student anticipate and plan for problems? Is the student comfortable using a trial and error method?
Reusing and remixing	Is the student efficient in researching relevant information? Can they use research to their advantage while maintaining authenticity? Can the student embed others' work into their own in a meaningful way? Does the student have critical code reading ability?
Modularizing	Can the student put together smaller parts to make something larger? Can the student piece together parts of a solution to solve a problem?

Note: Categories of skills were informed by Brennan and Resnick (2012) and Csizmadia et al. (2015)

Conclusion

The demand for CT to be integrated into elementary education is clear. What that looks like in terms of curriculum, practice, and assessment is not well defined. The data analysis process itself that was used in this research was an example of utilizing computational thinking and coding processes to analyze an expansive set of information such as the Ontario Curriculum documents. This chapter provides a preliminary analysis of one provincial-level elementary curriculum and recognizes the existence of computational thinking and related terms across disciplines. For example, there is an emphasis on CT-related concepts and perspectives in seemingly unrelated disciplines of language and the arts, with fewer instances of specific practices necessary for computer programming as evidenced by the low prevalence of key terms. Computational thinking is present in a variety of forms and contexts in the existing curriculum as both content (i.e., curriculum expectations) and pedagogical approaches (in planning, teaching strategies, and assessment). Educators therefore need to build on current expectations in each discipline to further develop CT as a way of thinking from elementary education onward. This chapter acknowledges the importance of identifying and defining CT as a metacognitive thinking process that teachers assess in collaboration with students. A set of questions is proposed to allow teachers and students to communicate the development of problem-solving skills across disciplines and developmental stages, serving as a foundational assessment tool for measuring CT in instruction and research.

Acknowledgments The research discussed in this chapter is supported by a Social Science and Humanities Research Council Insight Development Grant.

References

Ahonen, A. K. & Kankaanranta, M. (2015). Introducing assessment tools for 21st century skills in Finland. In P. Griffin & E. Care (Eds.), *Assessment and teaching of 21st century skills* (pp. 213–225). Springer.

Andrade, H. L. (2009). Students as the definitive source of formative assessment: Academic self-assessment and the self-regulation of learning. In H. L. Andrade & G. J. Cizek (Eds.), *Handbook of formative assessment* (pp. 90–105). New York, NY: Taylor & Francis.

Angelo, T. A., & Cross, P. (1993). *Classroom assessment techniques: A handbook for college teachers* (2nd ed.). San Francisco, CA: Jossey-Bass.

Barr, V., & Stephenson, C. (2011). Bring computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Transaction on Computational Logic, 2*(1), 48–54.

Bellanca, J., & Brandt, R. (Eds.). (2010). *21st century skills. Rethinking how students learn*. Bloomington, IN: Solution Tree Press.

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

Black, P., Harrison, C., Lee, C., Marshall, B., & Wiliam, D. (2003). *Assessment for learning: Putting it into practice*. New York, NY: Open University Press.

Black, P., & Wiliam, D. (2009). Developing the theory of formative assessment. *Educational Assessment, Evaluation and Accountability, 21*, 5–31. doi:10.1007/s11092-008-9068-5.

Brookhart, S. M. (2009). Mixing it up: Combining sources of classroom achievement information for formative and summative purposes. In H. L. Andrade & G. J. Cizek (Eds.), *Handbook of formative assessment* (pp. 279–296). New York, NY: Taylor & Francis.

Cashin, W. E. (1990). Students do rate different academic fields differently. *New Directions for Teaching and Learning, 43*, 113–121. doi:10.1002/tl.37219904310.

Charlton, P., & Luckin, R. (2012). Computational thinking and computer science in schools. In *'What The Research Says' Briefing 2*. The London Knowledge Lab, Institute of Education: United Kingdom.

Computer Science Teachers Association (CSTA), & The International Society for Technology in Education (ISTE). (2011). *Computational Thinking. Teacher Resources* (2nd ed.). USA: National Science Foundation. Grant No. CNS-1030054.

Constable, R. L. (2005). *Transforming the academy: Knowledge formation in the age of digital information*. New York, NY: Cornell University Publishing.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). *Computing at School*. Retrieved from http://www.computingatschool.org.uk

Dann, R. (2014). Assessment as learning: Blurring the boundaries of assessment and learning for theory, policy, and practice. *Assessment in Education, 21*(2), 149–166. doi:10.1080/0969594X.2014.898128.

Dede, C. (2010). Comparing frameworks for 21st century skills. In J. Bellanca & R. Brandt (Eds.), *21st century skills: Rethinking how students learn*. Bloomington, IN: Solution Tree Press.

DiCerbo, K. (2014). Review of the book chapter Assessment and teaching of 21st century skills, by P. Griffin, E. Care, & B. McGaw, Eds. *Assessment in education: Principles, policies, and practices, 21l*(4), 502-505. doi:10.1080/0969594X.2014.931836

Drake, S. M., Reid, J. L., & Kolohan, W. (2014). Interweaving curriculum and assessment in the 21st century: Engaging students in 21st century leanring. Toronto, ON: Oxford University Press.

Earl, L. (2003). *Assessment as learning*. Thousand Oaks, CA: Corwin Press.

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data*. Cambridge, MA: MIT Press.

Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive-developmental inquiry. *American Psychologist, 34*(10), 906–911. doi:10.1037/0003-066X.34.10.906.

Goyvaerts, J. (2016). *Regular-expressions*. Retrieved from: http://www.regular-expressions.info/index.html

Griffin, P. (2014). Performance assessment of higher order thinking. *Journal of Applied Measurement, 15*(1), 1–16.

Griffin, P., & Care, E. (Eds.). (2015). *Assessment and Teaching of 21st Century Skills: Methods and Approach*. Dordrecht: Springer.

Grover, S. (2015). *"Systems of Assessments" for deeper learning of computational thinking in K-12*. Draft paper presented at the Annual Meeting of the American Educational Research Association, Chicago, April 15–20, 2015.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Hattie, J. (2012). *Visible learning for teachers*. New York, NY: Routledge.

Hennessey, E., Mueller, J., Beckett, D., & Fisher, P. *Hiding in plain sight: Assessing how much and where computational thinking is represented with the ontario elementary school curriculum*. Unpublished manuscript.

Hesse, F., Care, E., Buder, J., Sassenberg, K., & Griffin, P. (2015). A framework for teachable collaborative problem solving skills. In *Assessment and teaching of 21st century skills* (pp. 37–56). Dordrecht: Springer.

Jun, S. J., Han, S. G., Kim, H. C., & Lee, W. G. (2014). Assessing the computational literacy of elementary students on a national level in Korea. *Educational Assessment, Evaluation and Accountability, 26*, 319–332. doi:10.1007/s11092-013-9185-7.

International Society for Technology in Education (ISTE). (2011). Retrieved from www.iste.org/computational-thinking

Leighton, J. P., & Gierl, M. J. (Eds.). (2007). *Cognitive diagnostic assessment for education: Theory and applications*. Cambridge, MA: Cambridge University Press.

National Research Council. (2010). *Report of a Workshop on the Scope and Nature of Computational Thinking*. Washington, DC: The National Academies Press. doi:10.17226/12840.

Naughton, J. (2012). Why all our kids should be taught how to code. *Observer New Review, 31*(3), 12.

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proceedings of the 41st ACM technical symposium on Computer science education: ACM, 265-269*.

Scratch Ed (2016). *Computational thinking with scratch*. Retrieved February 12, 2016, from http://scratched.gse.harvard.edu/ct/defining.html

Sadler, D. R. (1989). Formative assessment and the design of instructional systems. *Instructional Science, 18*(2), 119–144.

Shear, M. D. (2016). *Obama's budget urges a deeper commitment to computer education*. *New York Times*. Retrieved from http://www.nytimes.com/2016/01/31/us/politics/obamas-budget-urges-a-deeper-commitment-to-computer-education.html?_r=0

Shelby, C. & Woollard, J. (2013). Computational thinking: The developing definition. Available: http://eprints.soton.ac.uk/356481

Sternberg, R. J. (1986). *Intelligence applied*. New York, NY: Harcourt Brace Jovanovich.

Sternberg, R. J. (2009). Foreword. In D. J. Hacker, J. Dunlosky, & A. C. Graesser (Eds.), *Handbook on metacognition in education* (pp. viii–viix). New York, NY: Routledge.

Volante, L., & Beckett, D. (2011). Formative assessment and the contemporary classroom: Synergies and tensions between research and practice. *Canadian Journal in Education, 34*(2), 239–255.

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*, 715–728.

Voskoglou, M. G., & Buckley, S. (2012). Problem solving and computers in a learning environment. *Egyptian Computer Science Journal, ECS, 36*(4), 28–46.

Voskoglou, M. G., & Perdikaris, S. C. (1993). Measuring problem-solving skills. *International Journal of Mathematical Education in Science and Technology, 24*(3), 443–447.

Western and Northern Protocol for Collaboration in Education. (2006). *Rethinking classroom assessment with a purpose in mind*. Retrieved from http://www.wncp.ca/media/40539/rethink.pdf

Wilson, M., Scaliseb, K., & Gochyyev, P. (2015). Rethinking ICT literacy: From computer skills to social network settings. *Thinking Skills and Creativity, 18*(1), 65–80.

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

Wing, J. (2014). Computational thinking benefits society. *Social Issues in Computing Blog*. Retrieved from: http://socialissues.cs.toronto.edu/2014/01/computational-thinking/