| | |
|---|---|
| Title | Frame shifting as a challenge to integrating computational thinking in secondary mathematics education |
| Authors | Wendy Huang, Shiau Wei Chan and Chee Kit Looi |
| Source | *SIGCSE '21: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 390–396). |
| Organised by | Association for Computing Machinery, New York, United States |

# Frame Shifting as a Challenge to Integrating Computational Thinking in Secondary Mathematics Education

Wendy Huang
National Institute of Education
Nanyang Technological University
Republic of Singapore
wendy.huang@nie.edu.sg

Shiau Wei Chan
National Institute of Education
Nanyang Technological University
Republic of Singapore
shiauwei.chan@nie.edu.sg

Chee Kit Looi
National Institute of Education
Nanyang Technological University
Republic of Singapore
cheekit.looi@nie.edu.sg

## ABSTRACT

In this study, we adapted the notion of framing, a theoretical construct that refers to a person's expectations about social spaces (Goffman, 1974), to investigate whether teachers viewed computational thinking (CT) according to *subject-specific frames*. This case study aimed to understand how teachers make connections between CT and subjects targeted for integration. Epistemological framing contributed new insights on why teachers connected CT in different ways to different subjects: *frame shifting* focused teachers' attention on goals and activities specific to each subject. As teachers attended to a subject's particularities, they drew upon different *epistemic resources* to construct their descriptions of CT. Our participants (n=6) were teachers who taught both 7th-12th grade computing and mathematics as separate subjects. Qualitative coding of interview transcripts revealed that teachers' ideas about CT in computing were strongly influenced by computer programming while their ideas about CT in mathematics corresponded with familiar ways of teaching and learning mathematics. Instead of accepting fragmented notions of CT as the price of integration into individual subjects, we propose limiting the scope when defining CT. We explain how this non-intuitive strategy can preserve the coherence of CT and how it might be used in CT professional development (PD) for mathematics teachers.

## CCS CONCEPTS

• Social and professional topics~Computing education • Social and professional topics~Computational thinking • Social and professional topics~K-12 education


*SIGCSE '21*, March 13–20, 2021, Virtual Event, USA.

ACM ISBN 978-1-4503-8062-1/21/03...$15.00.
DOI: https://doi.org/10.1145/3408877.3432400

## KEYWORDS

Computational thinking; secondary mathematics education; teacher perspective; professional development; disciplinarity; theory; framing; epistemic resource

**ACM Reference format:**

Wendy Huang, Shiau Wei Chan, Chee Kit Looi. 2020. Frame Shifting as a Challenge to Integrating Computational Thinking in Secondary Mathematics Education. In *Proceedings of 2021 ACM Technical Symposium on Computer Science Education (SIGCSE '21), March 13–20, Virtual Event, USA*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3408877.3432400

## 1 Introduction

To embed computational thinking (CT) into K-12 teaching and learning, teachers ought to be well-prepared to facilitate students' use of CT's main ideas and dispositions to address interdisciplinary and discipline-specific problems [21]. There is growing attention among mathematics educators regarding the potential of CT in enhancing the comprehension of K-12 students on how and where mathematics matters [13]. For example, Lockwood, DeJarnette, and Thomas (2019) argued that "computing can and should be considered as a mathematical disciplinary practice" (p. 4). Nevertheless, there is inadequate information on how teachers understand CT concepts as they apply them in classroom practice [18]. Not only that, there are still questions about the types of teacher knowledge required to implement CT instruction and how to provide scaffolding to assist teachers to integrate CT into K-12 subject fields [16].

The literature on teacher professional development (PD) in CT has primarily reported positive feedback from the participants, including developing knowledge, confidence, materials, and desire to integrate CT into teaching [e.g., 1, 7, 8]. However, teachers have also consistently named barriers to implementation, including access to technology, logistics, scheduling, and support from school leaders [e.g., 11, 14]. We theorize that unnoticed epistemological obstructions may have also contributed to failures to integrate CT in teaching or to sustain such efforts.

The challenge is acute for teachers who specialize in single subjects, such as mathematics. These teachers typically hold degrees in the subjects they teach, and possess well-defined frames

based on disciplinary boundaries. The instructional goals and activities of subject-specific frames may play a significant role in how they interpret CT in the context of their subject. Since this process is largely subconscious, it would explain the gap between what teachers self-report following PD, and what they actually think or act upon.

Our study sought to advance theory on how teachers construct knowledge of CT and its connections to mathematics, and other school subjects more generally. We conducted semi-structured in-depth interviews of six teachers who taught secondary school (grades 7-10) and junior college (grades 11-12) mathematics and computing subjects, to see how they described and compared CT in these two subjects. Our research questions for this study were:

1. To what extent did terms that described CT mean different things in mathematics versus in computing?
2. How did subject-specific frames influence how teachers constructed their descriptions of CT?

## 2 Related Work

People cognitively activate frames when they attend to contextual cues that influence how they understand things. Epistemological framing is defined as "participants' understanding of kinds of knowledge that are relevant for use in their activity and the kinds of knowledge, understanding, and information they need to construct to succeed in their activity" [17]. Frames trigger which epistemic resources that people use to construct knowledge, to decide how and why they know something, whether consciously or not. We posit that teachers select epistemic resources about CT that are aligned with the instructional goals and content of the subject frame that is cognitively active at any given moment.

Teacher PD on CT typically offers teachers two forms of epistemic resources: (1) lists or frameworks of thinking skills, concepts, dispositions, and practices and (2) activities such as programming, robotics, or computer modelling. The former emphasizes the "thinking" of CT, while the latter emphasizes the "computational".

An example of "CT-as-thinking" came from Yadav, Stephenson, and Hong [21], who stated that the teachers should be provided with the chance to think computationally and experience CT as a generic set of capabilities and skills that did not require computers or other educational technology. They described CT as "… breaking down a difficult problem into more familiar ones that we can solve (problem decomposition), using a set of rules to find solutions (algorithms), and using abstractions to generalize those solutions to similar problems" (p. 570). However, a study done by Sands, Yadav, and Good [18] indicated that most of the teachers perceived that CT involved utilizing computers, using programming or coding, and employing technology in teaching. We name this second set of epistemic resources "CT-as-technology".

Weintrop et al. [19] and Lee and Malyn-Smith [9] analyzed the complex work of STEM professionals as the basis of their respective STEM+CT frameworks that promoted CT practices such as computational modelling and data analysis. Their work contributed to both forms of epistemic resources. The framework by Weintrop et al. [19] has been particularly influential in PD design for mathematics and science teachers. However, researchers who have studied science teachers' understanding of CT following PD experiences have reported concerns. Ketehult et al. [8] described teachers' blurring of engineering design and scientific inquiry with CT practices as evidence of possessing "an immature CT definition" (p. 185). Hestness et al. [6] warned that teachers who recognized CT in their existing teaching may not "incorporate it in new or expanded ways" (p. 431). As a result, the authors recommended that PD should help teachers appreciate the distinctive characteristics of CT relative to other kinds of thinking.

To illustrate how a problem could be solve using CT compared with mathematical deduction, let us consider the "lockers problem", which states that there are 1000 lockers and 1000 students; the first student opens every locker; the 2nd student closes every 2 lockers; the 3rd student opens every 3 lockers; each subsequent student alternates changing the state of locker doors until all students have done so. How many lockers remain open at the end?" This problem can be solved deductively with the insight that every locker that is a square number has an odd number of divisors while every non-square number has an even number of divisors. Therefore, non-square number lockers will be open/shut by pairs of students (and remain shut at the end) while square number lockers will remain open because there is no paired student to shut it. Contrast the deductive logic with a computational approach: Using CT, the problem could be formulated as a process where a student-agent visits each 1000 lockers per iteration, for 1000 iterations. A set of precise instructions could be devised for the agent to simulate opening and closing locker doors to find the solution. While the two approaches share some thought processes (e.g., abstracting away the surface details of lockers and students to a set of numbers), PD could focus teachers' attention on the differences, to sharpen teachers' understanding of the essence of CT.

## 3 Method

Our collaborators in the Singapore Ministry of Education connected us to the six teachers, according to our criteria: (1) each taught both mathematics and computing subjects, and (2) they collectively covered all grades 7-12 across a variety of schools (i.e., only two teachers were in the same school). The teachers had between 4 to 12 years of teaching mathematics, and 1 to 3 years of teaching computing. Teachers learned about CT from a variety of sources related to teaching computing, including a year-long course on the Python programming language, online courses, websites, articles, and meetings with other computing teachers. They did not mention participating in PD about CT in mathematics. All teacher participants had at minimum a bachelor's degree: three teachers had degrees in mathematics, two teachers had degrees in computer engineering, and one teacher had a PhD in computer science.

Each interview began by having the teacher read and comment on a definition of CT based on the Singapore computing syllabus, which describes CT as a problem-solving approach that

involves abstraction, decomposition, pattern recognition, and algorithmic thinking. The definition established a common vocabulary to talk about CT. The researchers proceeded to ask seven prepared questions that explored what the teachers thought CT meant in the two subjects and asked them to provide concrete examples from their teaching. Each audio-recorded interview lasted approximately 1 hour. Following each interview, the researchers wrote field notes with preliminary interpretations. The recordings were transcribed and imported into NVivo software for qualitative analysis. Table 1 contained brief definitions of each primary code, mainly derived from the textbook used in the computing subject. Additional definitions were added as needed to accommodate descriptions that appeared when teachers talked about CT in mathematics.

**Table 1. Definition of primary CT codes**

| Code (primary) | Definition |
|---|---|
| Decomposition | Breaking down a complex problem or process into smaller and more manageable parts (sub-problems) |
| Pattern recognition | Identifying similarities or common elements among two or more items |
| Abstraction | Identifying essential and relevant parts needed to solve a problem. Hiding details so that lower levels can be treated as black boxes. Generalizing a pattern. |
| Algorithmic thinking | Step-by-step instructions to express a process or solve a problem. |

Using the codebook, the first author coded an excerpt using "units of meaning"—any portion of text that corresponded to a code, regardless of length [5, 12]. Then she passed a copy with the marked portions but without codes to the second author to re-code. The codebook was revised to account for how teachers described CT in mathematics. Another excerpt was coded and compared between the two researchers. This was repeated with a complete transcript and the agreement reached an acceptable threshold of 0.59 using the formula for percent agreement for two raters [10]. A limitation of the study was that "units of meaning" had been criticized for being too subjective and leading for the second coder, thus often resulting in higher than typical inter-rater reliability scores. However, this strategy was suitable for studies where one researcher was more knowledgeable about the topic and there were limited human resources [3].

The first author organized the findings into tables with codes in the first column, followed by columns for each teacher. Each teacher's coded utterances were given short labels and each label was associated with a brief portion of the teachers' own words. The labels helped to identify themes among the various teachers' coded utterances. The first author then used the tables to surface findings that could be supported by the evidence. This paper presents one finding and discusses its significance.

## 4 Findings

Teachers' ideas about CT in computing were strongly influenced by computer programming while their ideas about CT in mathematics corresponded with familiar ways of teaching and learning mathematics.

### 4.1 The Meaning of CT in Computing

Teachers did not equate CT with programming. They distinguished between CT as the thought process of planning an algorithmic solution and coding as the act of translating a plan into a particular computer language syntax. However, the interviews revealed that most of the teachers drew heavily on programming constructs when describing the meaning of CT in computing. Teachers' answers to how CT was present in learning computing were fairly consistent and well defined.

Teachers' ideas about decomposition were informed by the goal of managing complexity, which could be done in several ways: (1) reducing the problem to the simplest form and "slowly build up to meet the requirement", (2) breaking down the problem into "cases" which could be handled separately, (3) breaking down the problem into increasingly smaller parts to make it easier to specify them precisely.

Pattern recognition included noticing changes in constants that could be generalized as variables and finding patterns in algorithmic execution that could be represented as loops. Besides being used interchangeably with generalization, abstraction also described "layers", with lower layers of procedures or functions treated as black boxes. Teachers mentioned how Python provided functions that students could use without knowing the details of how they work. One teacher also said that she wanted students to get underneath the abstraction by writing code using lower-level primitives. She believed that this would help students appreciate the power and importance of building layers of abstraction when designing computational solutions. One teacher said that in object-oriented programming, the data and its associated methods would be encapsulated as a form of abstraction.

Teachers agreed that constructing algorithms for machines required a high level of specificity, but wondered whether humans' execution of algorithms might be more fluid. Several teachers commented that students with higher mathematical ability may be able to "skip some steps" when performing mathematical algorithms—meaning that they could combine several steps in their heads—while machines and mathematically "weaker" students needed to follow steps that were more finely specified. Teachers also described how algorithms could be represented using structured English, flowcharts, pseudocode, and code. Table 2 summarized teachers' notions of CT in computing using the codes, labels to group related ideas, and examples.

**Table 2. Codes related to CT in computing with examples**

| Primary-secondary codes (# instances) | Examples |
|---|---|
| Decomposition-computing (12 instances) | managing complexity<br>• "simplify it such that it's easier to manage"<br>• "you will need to break it into cases"<br>• "breaking down the question into smaller chunks" |
| Pattern Recognition-computing (10 instances) | noticing what repeats, what changes<br>• "we get them to recognise the patterns of the... changes in the variables"<br>• "sometimes if a piece of code is being used over and over again...are they able to abstract it out"<br>• "you can use a loop for this repetition...what is it that...when you repeat it's still the same? And what is it when you repeat, it has to change?...change by how much?" |
| Abstraction-computing (13 instances) | generalization<br>• "so instead of the program dealing with 1 specific value?..You can just now use a variable to store whatever that you want"<br>black box<br>• "like for Python, they...have certain functions that already [are] preset, for example, ...print, will basically output the thing. But print is...a function that has an underlying code behind it."<br>hiding details<br>• "you're trying to...write a particular data type into a structure, and then you know, you have all the associated methods with it."<br>layers<br>• "So they will realise that len, it's to calculate length...And this is the first layer of abstraction. And then as we go on, len is being used in other functions...And, and that's where the second layer of abstraction comes in. So this has been abstracted layer by layer and students starts to see that...eventually I don't really care that much about the trivial anymore." |
| Algorithmic thinking-computing (25 instances) | • process: "So what's the thing that is missing in between, that's to think about." (to construct)<br>high specificity<br>• "for computing there are...a bit more levels in a sense, the part 1, part 2, part 3, but...in between the parts, there are sub-parts"<br>• "the algorithmic thinking of the human being can be slightly fluid, but ultimately has to translate to something more rigid in order for the computer to actually do"<br>concepts<br>• loops, decisions<br>representations<br>• diagrams<br>• structured English<br>• flowchart<br>• pseudocode |

## 4.2 The Meaning of CT in Mathematics

Although teachers did not explicitly report using any CT terminology when teaching mathematics, they described how CT was implicitly taught or used to learn topics including numbers, algebra, geometry, probability, and calculus. For decomposition, teachers discussed how to "break down" problems, objects, or concepts into parts. One could break down problems into steps (to solve the problem), sub-problems (to solve separately), or key words (to identify relevant details). Also, objects like numbers could be broken down into prime numbers or decomposed using number bonds. Pattern recognition involved identifying common features, noticing repetition in sequences, and observing relationships between mathematical components (e.g., direct proportions, even laws of indices). These patterns and relationships could be generalized into abstractions such as rules or formulas. Teachers described algorithmic thinking as following step-by-step instructions and expressing one's thinking precisely and logically so that another person could follow the same process.
Table 3 summarized teachers' descriptions of CT in mathematics using the codes, labels to group related ideas, and examples.

**Table 3. Codes related to CT in mathematics with examples**

| Primary-secondary codes (# instances) | Examples |
|---|---|
| Decomposition-mathematics (18 instances) | breaking down<br>• objects: "a big number can be broken down to smaller prime numbers" |

| | |
|---|---|
| | • concepts or terminology: "highest common factor...I would break down as in highest, what does highest mean? common, why is it common? factor, what is a factor?"<br>• problems into steps: "I know my start, I know my end. I can do things like how do I work backwards"<br>• problems into sub-problems: "there's some questions where it's already broken down to Part A, Part B, Part C, Part D."<br>• problems into key words: "you decompose the problem, then you make sure that the kids find keywords" |
| Pattern recognition-mathematics (18 instances) | closely related to abstraction or generalisation<br>• "if you see a pattern, you can sort of generalise something"<br>identifying common features<br>• "I erase only numbers that are related to the question that we did before. And then after that...you see, this is the same pattern"<br>Curriculum topics: number patterns/sequences, direct proportion |
| Abstraction-mathematics (26 instances) | rules / formulas<br>• "all formulas are abstraction of a certain pattern"<br>efficient problem-solving: template, cases, relevant details, structure<br>• "if the question asks for something, usually...the question has a certain structure, usually the template is the same just that the numbers change."<br>black box (two contradictory notions)<br>• "you may not know why something happens, but you know how it happens"<br>• "The only very relevant abstraction in Math that I would see is probably in the topic of functions. You're not particularly or entirely sure of how, what, what happens underlying, but you know how to use it, and what kinds of output you will want to expect" |
| Algorithmic thinking-mathematics (47 instances) | performing computations or other mathematical procedures<br>• "in primary school, let's say you teach computation on paper, that's an algorithm, right?" |
| | • "it's mainly the algebra topics, where you have these sorts of processes that you want the students to learn."<br>communicating clearly and logically<br>• "to be able to articulate it clearly and in a very step by step format, so that others would be able to understand what they're trying to do."<br>process to get a solution<br>• "what are the different steps that [you] actually has to go through before you get your so called output"<br>systematic thought process<br>• "if you see the situation, do this, if you don't, then try something else." |

## 5 Discussion

Based on the evidence, we briefly answer the research questions as follows: (1) teachers described CT in different ways in the two subjects but perceived coherence in their understanding of CT, (2) subject-specific activities, concepts, and purposes shaped the descriptions that teachers constructed of CT. In this section, we use frame shifting and the subject-specific activation of epistemic resources to explain these claims.

Depending on the subject frame, teachers perceived problem-solving differently. In computing, students "solved" a programming problem by specifying an information processes that met specifications. The solution was often expressed in code so that a "dumb" agent, whether human or machine, could blindly follow the instructions without needing to understand or make any judgments. In computing, the set of instructions is the solution and is validated using test cases. However, in a mathematics frame, students solve a math problem by reasoning with appropriate conceptual and procedural knowledge to compute an "answer" (Henderson, 2003). In school mathematics, the computed output is the solution and is validated by "workings" that demonstrate a sequence of deductive reasoning.

Teachers perceived an important role for algorithmic thinking in both subjects, but for different reasons. In computing, algorithmic thinking primarily involved the construction of an algorithm, the "process" step of the problem formulation "input-process-output". Teachers' descriptions of decomposition, pattern recognition, and abstraction showed how these thinking skills supported the creative act of constructing an algorithm. However, teachers saw the purpose of algorithmic thinking in mathematics as gaining procedural fluency or communicating one's thinking clearly. Rich, Yadav, and Schwarz [16] asserted that many algorithms in mathematics were traditional and standardized techniques to execute arithmetic computations. In the words of one interviewee: "There's a lot of following algorithms, but creating algorithms, not so much." However, teachers expressed not only wanting students to correctly follow mathematical procedures, but also wanting students to understand the

concepts embedded in the procedures. For this reason, teachers' descriptions of decomposition, pattern recognition, and abstraction showed how these thinking skills were used to scaffold understanding concepts.

Table 4 summarized the effect of subject-based frames on two CT themes.

**Table 4: Contrasting CT themes in computing and mathematics subjects**

| | Computing frame | Mathematics frame |
|---|---|---|
| Problem solving | solution is the computing process | solution is the computed output (through reasoning) |
| Algorithmic thinking | creating an algorithm | following an algorithm (with understanding) |

Subject-based framing shifts helped us understand how the mental discrepancies occurred, but not why. To answer this question, we consider which *epistemic resources* were selected by teachers as they constructed their descriptions. When the mathematics frame was activated, teachers seemed to draw upon the "CT-as-thinking" resource, which conflated CT with familiar ways of teaching and learning in mathematics. When describing CT in computing, teachers relied on the "CT-as-technology" resource, which constrained CT to the activity of programming. Framing and epistemic resources worked together to explain the cognitive mechanism that resulted in a fragmented perception of CT.

An interesting puzzle was that the apparent dissonance in the meanings of CT across different subjects did not seem to discomfort teachers. Teachers not only recognized elements of CT in both subjects, but also perceived coherence. Take the term "decomposition": the differences in what and why things were "decomposed" in mathematics and computing subjects were not significant as long as the acts shared the general idea of "breaking down into parts".

Some might ask: what is wrong with using CT to serve different goals? We argue that both a fragmented and an overly general understanding of CT weakens its purpose as a conceptual tool for tackling some problems better than others. According to Rambally [15], while CT practices such as "problem representation, abstraction, problem decomposition, recursion, modeling, simulation, verification, and prediction" share elements with mathematical thinking (MT), CT "extends MT skills in a unique way" (p. 417). He cited Denning et al. [4], who called attention to the computing paradigm's distinctive focus on information processing.

Denning's stance aligns with an often used definition of CT, that it is "the thought process involved in formulating problems and developing approaches to solving them in a manner that can be effectively carried out by an information processing agent (whether machine or human role-playing one)", based on Wing's definition [20]. The thinking skills listed in this paper, among others in literature and standards, should be facets that work together to support that central goal.

We propose using Wing's simple definition of CT for integration in mathematics to preserve balance and coherence regarding the meaning of CT. While such definition may appear too constrained when compared to the expansive STEM framework developed by Weintrop et al. [19], the CT Integration practices by Lee and Malyn-Smith [9], or the ScratchMaths 5E framework [2], we believe that it addresses a core concern of mathematics instruction: problem-solving.

We recommend the following for CT PD for mathematics teachers:

- Highlight CT as a specific kind of problem-solving approach where the solution is a set of instructions for information processing agents to carry out, not a general problem-solving strategy.
- Have teachers solve problems using both computational and non-computational methods. Reflect on the pros and cons. Develop a list of characteristics to identify problems that are well suited for CT.
- Give teachers opportunities to construct computational solutions using a variety of representations such as structured natural language, flowcharts, pseudocode, or novice-friendly programming languages.

## 6 Conclusion

Widespread public support for students to learn CT and coding has led to increased global demand for answers on how to bring CT into an overstuffed school curriculum. This study reported a finding that teachers' speculations about connections between CT and mathematics differed from their ideas of CT in computing; yet teachers still perceived a coherent CT. We proposed the following analytic generalization to explain the finding: as teachers shifted from one subject frame to another, they selected epistemic resources of CT that fit the goals and content of the subject, to construct descriptions of connections between CT and the framed subject. The subconscious nature of the process prevented teachers from recognizing that they were conflating CT with other thinking, or possessed a narrow association of CT with programming.

To avoid these extremes, PD should introduce teachers to just one epistemic resource as a starting point. The operationalized definition can include programming but would not be coupled with it. Frames are fluid; as teachers become comfortable with CT, its limited definition can be expanded upon to include other computing paradigms (e.g., machine learning) and disciplinary-relevant practices (e.g., data science, modelling).

## ACKNOWLEDGMENTS

This work was supported by a ERFP grant from the Ministry of Education under Grant Number OER 10/18 LCK. We thank our colleague Dr. Michael Tan for his valuable feedback.

## REFERENCES

[1] Ahamed, S.I. et al. 2010. Computational thinking for the sciences: a three day workshop for high school science teachers. Proceedings of the 41st ACM technical symposium on Computer science education (New York, NY, USA, Mar. 2010), 42–46.

[2] Benton, L. et al. 2017. Bridging Primary Programming and Mathematics: Some Findings of Design Research in England. Digital Experiences in Mathematics Education. 3, 2 (Aug. 2017), 115–138.

[3] Campbell, J.L. et al. 2013. Coding In-depth Semi-structured Interviews: Problems of Unitization and Intercoder Reliability and Agreement. Sociological Methods & Research. 42, 3 (2013), 294–320.

[4] Denning, P.J. and Freeman, P.A. 2009. The profession of IT: Computing's paradigm. Communications of the ACM. 52, 12 (Dec. 2009), 28–30.

[5] Garrison, D.R. et al. 2006. Revisiting methodological issues in transcript analysis: Negotiated coding and reliability. The Internet and Higher Education. 9, 1 (Jan. 2006), 1–8.

[6] Hestness, E. et al. 2018. Professional Knowledge Building within an Elementary Teacher Professional Development Experience on Computational Thinking in Science Education. Journal of Technology and Teacher Education. 26, 3 (Jul. 2018), 411–435.

[7] Jaipal-Jamani, K. and Angeli, C. 2017. Effect of Robotics on Elementary Pre-service Teachers' Self-Efficacy, Science Learning, and Computational Thinking. Journal of science education and technology. 26, 2 (Apr. 2017), 175–192.

[8] Ketelhut, D.J. et al. 2020. Teacher Change Following a Professional Development Experience in Integrating Computational Thinking into Elementary Science. Journal of science education and technology. 29, 1 (Feb. 2020), 174–188.

[9] Lee, I. and Malyn-Smith, J. 2020. Computational Thinking Integration Patterns Along the Framework Defining Computational Thinking from a Disciplinary perspective. Journal of science education and technology. 29, 1 (Feb. 2020), 9–18.

[10] Miles, M.B. and Huberman, A.M. 1984. Qualitative data analysis: a sourcebook of new methods. Sage Publications.

[11] Morreale, P. et al. 2012. Measuring the impact of computational thinking workshops on high school teachers. J. Comput. Sci. Coll. 27, 6 (Jun. 2012), 151–157.

[12] Morrissey, E.R. 1974. Sources of Error in the Coding of Questionnaire Data. Sociological methods & research. 3, 2 (Nov. 1974), 209–232.

[13] Pérez, A. 2018. A Framework for Computational Thinking Dispositions in Mathematics Education. Journal for Research in Mathematics Education. 49, 4 (2018), 424–461.

[14] Pollock, L. et al. 2017. From Professional Development to the Classroom: Findings from CS K-12 Teachers. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (New York, NY, USA, Mar. 2017), 477–482.

[15] Rambally, G. 2015. The Synergism of Mathematical Thinking and Computational Thinking. Cases on Technology Integration in Mathematics Education. IGI Global. 416–437.

[16] Rich, K.M. et al. 2019. Computational Thinking, Mathematics, and Science: Elementary Teachers' Perspectives on Integration. Journal of Technology and Teacher Education. 27, 2 (2019), 165–205.

[17] van de Sande, C.C. and Greeno, J.G. 2012. Achieving Alignment of Perspectival Framings in Problem-Solving Discourse. Journal of the Learning Sciences. 21, 1 (Jan. 2012), 1–44.

[18] Sands, P. et al. 2018. Computational Thinking in K-12: In-service Teacher Perceptions of Computational Thinking. Computational Thinking in the STEM Disciplines: Foundations and Research Highlights. M.S. Khine, ed. Springer International Publishing. 151–164.

[19] Weintrop, D. et al. 2016. Defining Computational Thinking for Mathematics and Science Classrooms. Journal of science education and technology. 25, 1 (Feb. 2016), 127–147.

[20] Wing, J.M. 2010. Computational Thinking: What and Why? The Link (CMU).

[21] Yadav, A. et al. 2017. Computational thinking for teacher education. Communications of the ACM. 60, 4 (Mar. 2017), 55–62.