Applied Measurement in Education

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/hame20

Formative Assessment of Computational Thinking: Cognitive and Metacognitive Processes

Sarah Bonner, Peggy Chen, Kristi Jones & Brandon Milonovich

To cite this article: Sarah Bonner, Peggy Chen, Kristi Jones & Brandon Milonovich (2021) Formative Assessment of Computational Thinking: Cognitive and Metacognitive Processes, Applied Measurement in Education, 34:1, 27-45, DOI: 10.1080/08957347.2020.1835912

To link to this article: https://doi.org/10.1080/08957347.2020.1835912

Published online: 01 Feb 2021.

APPLIED MEASUREMENT IN EDUCATION
2021, VOL. 34, NO. 1, 27–45
https://doi.org/10.1080/08957347.2020.1835912

Routledge
Taylor & Francis Group

Formative Assessment of Computational Thinking: Cognitive and Metacognitive Processes

Sarah Bonner, Peggy Chen, Kristi Jones, and Brandon Milonovich

ABSTRACT

We describe the use of think alouds to examine substantive processes involved in performance on a formative assessment of computational thinking (CT) designed to support self-regulated learning (SRL). Our task design model included three phases of work on a computational thinking problem: forethought, performance, and reflection. The cognitive processes of seven students who reported their thinking during all three phases were analyzed. Ratings of artifacts of code indicated the computational thinking problem was moderately difficult to solve (M = 15, SD = 5) on a scale of 0 to 21 points. Profiles were created to illustrate length and sequence of different types of cognitive processes during the think-aloud. Results provide construct validity evidence for the tasks as formative assessments of CT, elucidate the way learners at different levels of skill use SRL, shed light on the nature of computational thinking, and point out areas for improvement in assessment design.

The expansion of computer science (CS) education at the PreK-12 level has become a priority in the United States of America (USA) due to burgeoning demand for workers trained in computer and information technology (Department of Labor Bureau of Labor Statistics, 2020). Multiple PreK-12 curricula and courses have been developed and disseminated in recent years, several of which stress conceptual understanding over coding. Proponents of the conceptually based approach to CS education assert that a system of thinking underlies computing, and students need to be able to use this way of thinking to read and write in the language of computers (Román-González, Perez-González, & Jimenez-Fernandez, 2017). The construct *computational thinking* (CT), coined by Wing (2006), has been proposed to refer to mental processes that people use to represent problems as computational steps and algorithms (Aho, 2011). While a number of conceptually based curricula that draw on CT concepts have been developed and are now being taught in schools in the USA, the assessment of CT is still in its infancy. This is especially true for formative assessment (FA). To extend the potential of PreK-12 CS education to benefit all students, CS teachers need access to FA materials that are supported by evidence of technical quality and have a sound basis in learning theory (Pellegrino, Chudowsky, & Glaser, 2001).

This study was part of a larger project of development of FA tasks in computational thinking, and research on their use to advance the assessment *for* learning purpose of FA: activation of students' learning through assessment of concepts and skills (Wiliam & Thompson, 2008). We developed the tasks according to a cognitive model of self-regulated learning (SRL). SRL is a goal-directed process that consists of analyzing tasks; deploying strategies; monitoring progress; seeking resources when needed; and adjusting tactics, if necessary, to generate a solution. SRL was an appropriate model for FA design because of emerging theory linking SRL to FA (Andrade & Brookhart, 2016; Chen & Bonner, 2020). With SRL as a design framework, we sought to leverage learning benefits demonstrated

CONTACT Sarah Bonner sbonner@hunter.cuny.edu Department of Educational Foundations and Counseling Programs Hunter College, City University of New York New York, New York 10065

both for FA (Kingston & Nash, 2011) and SRL (Chen & Cleary, 2009; DiGiacomo & Chen, 2016), to the end of boosting their combined potential to help students learn.

Because the design model for FA tasks in this study involved two complex concepts, SRL and CT, evaluation of task quality in terms of constructs was paramount. We therefore used think alouds as our primary approach to validation. Since the mid-twentieth century, cognitive psychologists and theorists of educational measurement have recommended that test design and interpretation be informed by study of the substantive processes at work in test taking as they relate to the constructs to be measured, using methods such as think alouds (Cronbach, 1971; Leighton, 2004; Messick, 1989).

In the present paper, we review essential components of FA, the CT construct, and theory of SRL. We briefly recount our design process for FA tasks, then describe our methods and present results of a study that used think alouds to examine student cognitive processes during performance on the FA tasks. This work presents information gleaned about the nature of the thinking processes demonstrated by students as they worked through FA tasks, and the alignment of those processes with the theoretical constructs of computational thinking and self-regulated learning theory. It holds implications for intentional FA task design that leverages overlapping processes among FA, SRL, and CT to achieve synergistic impacts on learning.

1. Literature Review

Multiple initiatives have been put forth to address the workforce need for computer scientists in the USA. In 2016 former President Obama promoted the expansion of CS PreK-12 education in the USA by releasing massive federal funding to states and school districts for CS education, and earmarking National Science Foundation funding for CS education. A consortium of organizations including the Computer Science Teachers Association (CSTA), Code.org, and the Association for Computing Machinery followed suit by producing a *K-12 Computer Science Framework* as a foundation for states, districts, and organizations in the USA to generate CS K-12 standards. Shortly thereafter, the CSTA published *CSTA K-12 Computer Science Standards*, which drew from concepts in the computational thinking (CT) literature to articulate learning objectives for CS curriculum and instruction at the K–12 level (Computer Science Teacher Association (CSTA), 2017). Along with these initiatives, PreK-12 CS course packages, applications, and modules were created, and disseminated rapidly. In secondary schools, several entirely new curricula for upper middle and secondary school stand-alone CS courses that emphasize CT were developed. School systems rose to the call for CS education by providing professional development for teachers. In New York City (NYC) alone, 1,900 teachers across 800 schools were trained in CS between 2016 and 2019 (New York City Department of Education, 2019).

Accompanying the spread of secondary school curricula, new summative assessments for CS courses were developed. Prior to the last decade, the only nationally distributed secondary school examination in CS in the USA was the Advanced Placement: CS Application (AP:CSA) examination, which primarily covers coding in Java script and has been in existence in some form since 1984. In the last decades, additional summative assessment linked to curricula that emphasize CT have been developed. Beginning in 2012, SRI International developed and has since disseminated four end-of-unit tests and a final examination for the early high school course Exploring Computer Science. In 2017, the College Board also began to offer an examination for the conceptually grounded approach to CS, the Advanced Placement Computer Science Principles test (AP:CSP). However, the development of formative assessment (FA) in CS has lagged behind the development of standards, curriculum, and tests.

1.1. Formative Assessment and Computer Science Education

Classroom-based FA, or assessment *for* learning, is assessment that generates information for the purpose of feedback and modifications to teaching and learning (Black & Wiliam, 1998). According to Bennett (2011), any task designed specifically to provide information to guide learning can be considered

FA. Other researchers take a process view: FA is a series of decisions and interactions between teachers and learners that motivate and direct learning (e.g., Shepard, Penuel, & Pellegrino, 2018). Some conceptualizations of FA include the premise that students should be active participants in the assessment process (Broadfoot et al., 2002; Wiliam & Thompson, 2008). Students can use FA information to activate and self-direct their own learning, thereby promoting the purpose of assessment *for* learning.

Resources for FA in CS at the secondary school level are scarce. There are a few published instruments, but these have largely been developed for specific programming environments and are targeted at elementary or middle-school students (e.g., Moreno-León & Robles, 2015; Werner, Denner, Campe, & Kawamoto, 2012). Few of these published sources provide information about how these instruments can be used for instructional decision-making within classrooms, and information on their technical quality is incomplete. Only recently has quality in classroom assessment become an active area of research in CS education (Grover et al., 2017; Román-González, Moreno-León, & Robles, 2017). Curricula associated with courses such as AP:CSP (e.g., University of California at Berkeley's Beauty and Joy of Computing, Code.org's CS Principles) contain tools like self-checking quizzes, scoring rubrics for projects, and questions embedded in the programming environment. However, these curriculum-based materials lack documentation about task quality. In practice, teachers' FA methods in CS classrooms in the USA range broadly, from questioning individual students, to observation, to classroom polling (Yadav et al., 2015). It is evident that CS education in the USA lacks practical tools for high-quality FA at the secondary school level, particularly for assessment of the complex cognitive construct of computational thinking (CT).

1.2. Computational Thinking

Computational thinking (CT), described as a "universally applicable attitude and skill set" (Wing, 2006, p. 33), is the key construct associated with most contemporary conceptually-based approaches to PreK-12 CS education. Experts have suggested many definitions and ways of operationalizing CT. Variability in their definitions suggests that the CT construct may be called emergent; the boundaries that demarcate what is and is not computational thinking are not fully agreed upon. Aho (2011) provides a useful general definition that articulates the distinctly computational nature of the construct: "the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms" (p. 832).

Shute, Sun, and Asbell-Clarke (2017) synthesized the CT literature and identified core facets of computational thinking: *abstraction, decomposition, algorithms, debugging, iteration, and generalization*. Computational thinkers demonstrate *abstraction* when they identify structural patterns beneath the surface features of a problem that can be used or combined to form a solution. Computational thinkers use *decomposition* when they break a complex problem into manageable steps, modularize its elements, or use top-down design. Computational thinkers use *algorithms* when they apply logical statements such as loops, conditionals, and functions to solve problems. Computational thinkers use *debugging* procedures such as testing to detect and resolve errors. They use *iteration* when they repeat the same or similar elements over multiple trials, with or without variations and modifications, to reach a solution. They practice *generalization* when they reuse or remix elements of successful strategies to solve problems of similar underlying structures (Brennan & Resnick, 2012; Grover & Pea, 2013; Shute et al., 2017). Brennan and Resnick (2012) usefully organize the CT construct into two areas: *concepts* and *practices*. CT *concepts* consist mainly of the algorithmic aspects of CT that are learned in different programming environments. CT *practices* are those facets of CT that thinkers use across environments, and include abstraction (combined with decomposition); testing and debugging; iteration; and generalization. Brennan and Resnick (2012) also discuss CT *perspectives*, which are outside the scope of the present study.

1.3. Self-Regulated Learning: A Theoretical Framework Compatible with FA and CT

The model for FA task design and validation of CT assessment in the present study was based on a social-cognitive approach to self-regulated learning (SRL) theory. This study drew on Zimmerman's (2000) three-phase model of SRL. According to Zimmerman (2002), SRL can be understood as "the self-directive process by which learners transform their mental abilities into academic skills" (p. 65). Zimmerman's model highlights the importance of task analysis, goal setting, control, monitoring, and self-evaluation as prominent self-regulatory processes for successful academic learning. The model involves cyclical and dynamic feedback processes that occur during three cyclical phases of self-regulation: forethought, performance, and self-reflection. Metacognitive monitoring and control are critical components of SRL. Learners exhibit metacognitive monitoring when they internalize information obtained through self-feedback and external sources to iteratively modify their strategies and actions (Zimmerman, 2013). They exhibit control when they initiate SRL processes, for instance, by critically analyzing a task, explicitly setting goals, or reflecting on teacher feedback to direct their behaviors toward achieving their goals (Zimmerman & Schunk, 2011).

The first phase of Zimmerman's model, forethought, focuses on goal setting and task analyses, when self-regulated students analyze the task at hand, plan learning strategies, and think about how they can break down a complicated task into manageable sub-tasks. In the second phase of SRL, performance, learners actively engage with a task or problem to be solved. During this phase, learners monitor their progress and use various strategies. Students may generate self-feedback during this phase of the SRL cycle to check their understanding and modify strategies. Monitoring one's learning process is considered desirable metacognitive functioning on the part of learners, and is exhibited with higher-performing students (Callan & Cleary, 2019). Monitoring learning progress may lead to help-seeking behavior or use of resources such as peers, teachers, and/or technology. The third phase of the model, reflection, centers on self-evaluation and adjustment to improve learning in the future. During this phase, students engage in evaluating their performance based on goals and standards. They may engage in attribution, which refers to what people believe to be the causes of their successes or failures, such as their ability, the amount of their effort, the difficulty of the task, and luck (Weiner, 2010). Students regularly make attributions to explain to themselves or others why they did poorly or well on tests, projects, and papers. Their perceived causes for the results influence their emotions and choices of subsequent actions.

Some learners purposefully initiate SRL processes to direct their behaviors, thinking, emotions, and environment to achieve desired goals (Zimmerman & Schunk, 2011). Empirical research has shown that self-regulated math students attain high performance (Chen & Cleary, 2009; DiGiacomo & Chen, 2016). Sophisticated learners exhibit heightened metacognitive awareness and engage in iterative modification of their learning, incorporating information obtained both through self-feedback and external sources (Zimmerman, 2013). This continuous personal feedback mechanism is vital to any successful SRL process, as it signals to learners the potential usefulness of information acquired from one phase, so they can adjust their plans and behavior during the next phases in the learning sequence. Research has also shown that SRL can be learned (DiBenedetto & Zimmerman, 2010; Mason, 2004).

SRL theory is particularly relevant to the present study because of conceptual arguments that SRL substantially overlaps with FA processes and practices (Andrade & Brookhart, 2016; Chen & Bonner, 2020), and that it can be a particularly useful approach for learning CT (Peters-Burton, Cleary, & Kitsantas, 2015). Paris and Paris (2001) perhaps first presented the connection between SRL processes and classroom assessment practices by examining trends and areas of research in SRL (e.g., literacy instruction, cognitive engagement, and self-assessment) that have direct applications for classroom instruction. Paris and Paris (2001) articulated the importance of student internalization of learning through active monitoring and controlling mechanisms, and through self-assessment that reinforces self-awareness of how and what was learned. Andrade and Brookhart (2016) pointed out that FA and SRL are both cyclical, and take a student-centered and process-oriented approach that affords learners opportunities to make multiple attempts to adjust their learning strategies to close gaps between what

they can do and what they want to achieve. FA and SRL also share the propelling mechanism of feedback during monitoring and reflection (Andrade and Brookhart, 2016). Peters-Burton et al. (2015) used the Zimmerman (2000) three-phase model to illustrate how students might learn and develop CT skills such as abstraction and pattern generalization with self-regulation. Conceptual overlaps among SRL, FA, and CT frameworks point to the possibility that formative tasks can be designed a) to combine assessment of CT with SRL prompts without interfering with CT-aligned thinking processes for test-takers, and b) take advantage of the synergies between SRL and FA to activate student thinking, to boost the impacts of learning associated with either SRL or FA alone (Cleary & Chen, 2009; DiGiacomo & Chen, 2016; Kingston & Nash, 2011).

1.4. Construct-Based Arguments Using the Think-Aloud Approach

The issue of construct validity is central to any argument about the technical quality of an assessment task (Messick, 1989). Before consideration of whether formative task-based information can be used to achieve intended impacts on teacher or student decision-making, it is essential to establish that interpretations derived from the tasks elicit thinking processes aligned with their intended constructs, in this case, CT and SRL. Therefore, construct validation of interpretations about student CT and SRL is the focus of the present study. To deduce whether students' cognitive processes in response to assessment tasks are consistent with constructs targeted for assessment, researchers often use think-aloud methods. Analysis of think alouds also "feeds back to knowledge of what it means to be an expert within the content domain" (Leighton, 2004, p. 8). The benefits of the think-aloud approach are at least twofold: think alouds can provide evidence of construct validity of test-based interpretations, and can lead to increased understanding of the psychological constructs underlying performance in a domain. Think alouds are appropriate for examining problem-solving processes at work in test performance, and confirming cognitive models (Leighton, 2017). Despite calls for new studies that incorporate response process evidence (Leighton, 2017; Zumbo & Hubley, 2017), relatively few validation studies devote substantial attention to the use of think alouds.

Cognitive process studies typically rely on examinees' verbal reports of their thinking concurrently with task performance. Tasks appropriate for think-aloud studies usually involve moderately complex problem solving, rather than simple knowledge or basic application or known problem-solving rules, such as remembering learned content. The latter types of thinking may be automatic for learners and therefore not accessible during thinking aloud (Leighton, 2017). Contemporary studies using the think-aloud method in assessment validation have focused both on problem-solving tasks and metacognitive assessment, and include research on the PISA assessment of self-efficacy (Pepper, Hodgen, Lamesoo, Kõiv, & Tolboom, 2018), science design tasks for elementary school students (Kelley, Capobianco, & Kaluf, 2015), and collaborative problem solving in computing (Siddiq & Scherer, 2017).

Questions have been raised about the quality of information obtained from think-aloud research, due to participant reactivity (Russo, Johnson, & Stephens, 1989; Wilson, 1994). A validation method is reactive if the use of the method systematically changes the way in which examinees perform. Reactivity can affect performance level, speed, or type of mental process. After reviewing multiple experimental and quasi-experimental studies, Ericsson and Simon (1993) concluded that performance level was not affected for verbal reports of mental processes when subjects verbalized concurrently with task performance without introspection. Concurrent verbalization tends to take more time, but time is only a concern with speeded assessments.

Questions remain about the relationship between the types of mental processes derived from think-aloud studies, compared to the types of mental processes that would be at play in test performance without thinking aloud. Subjects may find it difficult to retrieve their thinking accurately while speaking because of competition for space in working memory, or because their attention is divided between verbalizing and problem solving (Leighton, 2017). However, Nisbett and Wilson (1977) recognized that verbal reports are accurate in some contexts, particularly when they are elicited

close in time to the application of the stimulus. They stated, "reports will be accurate when influential stimuli are (a) available and (b) plausible causes of the response, and when (c) few or no plausible but noninfluential factors are available" (p. 253). Recent research (Fox, Ericsson, & Best, 2011; Leighton, 2017) tends to be less concerned with reactivity due to concurrent thinking aloud, in comparison to reactivity of other methods that are intended to reveal similar qualities of cognition. Leighton draws a distinction between cognitive labs, where the interview takes place after task completion, and concurrent think alouds. According to Leighton (2017), cognitive labs are more reactive than think alouds because participants respond to the prompts of the interviewer, rather than to the task itself.

The purpose of this study was to develop quality FA tasks that are theory-driven, and yield information that can be interpreted in terms of the targeted constructs. We elected to use the think-aloud approach to validation because think alouds probe into the cognitive processing of examinees during testing. This was highly appropriate for the type of task we developed, which involved CT problem solving and opportunities for students to plan, monitor, and reflect. The following research questions guided our study: 1) Do student think alouds provide evidence that computing problems in the FA task elicit cognitive processes consistent with conceptual definitions of CT? 2) Do student think alouds provide evidence that SRL prompts embedded in the FA task elicit cognitive processes consistent with SRL theory? 3) How do students at different levels of mastery vary in use of CT and SRL? 4) Do think alouds provide evidence that elucidates definitions of the CT construct or SRL processes?

2. Methods

2.1. Preliminary Work: Development and Content Validation

Prior to validating the FA tasks in terms of the cognitive processes they elicited, we used content validation approaches iteratively, throughout the design process. Formal think alouds were conducted and analyzed only when there was evidence that our task prototype appropriately represented our ideas about CT and SRL, while maintaining desirable qualities for classroom assessment such as reading levels suitable for diverse learners and feasibility of administration. Following Downing's (2006) steps for test development, the first and second authors set the overall plan for the work, including the constructs to be measured, purpose and desired interpretations of assessment, a model of task format, and approaches to validation. We then formed a design team that included a computer scientist, professional developer in CS education, two educational psychologists with specializations in SRL and assessment, and nine experienced CS teachers. These individuals, along with the theoretical and empirical literature, were our essential sources of content validity evidence. Together, the design team constructed the operational definitions of CT and SRL presented above, using not only the scholarly literature, but with reference to professional knowledge and CS curricula.

We proceeded to the development of stimuli or tasks. This was an iterative process as the team tried out various ways to concretize the operational definitions into actual tasks, and analyzed how well those tasks represented the constructs. During task development, the team members debated the alignment between the operational definitions of CT and SRL, and their representation in task content. After each task draft, one or more CS teachers would perform an informal trial, which typically included asking one or more students to provide feedback on their perceptions of task alignment to instruction, difficulty, and clarity. Following such a trial, the team debriefed using student work samples and feedback provided by students. This approach of using feedback from students as part of validation is appropriate for classroom assessment in that it helps assessment developers bear instructional validity in mind as well as content (Bonner, 2013). We then proceeded to use think alouds to understand the substantive cognitive and metacognitive processes that students used in response to the FA tasks, and compare those processes to the theoretical processes in CT and SRL which we intended to elicit for assessment (Cronbach, 1971; Leighton, 2004; Messick, 1989).

2.2. Study Context and Participants

Data collection for this study took place in three nonselective public schools and one charter school in NYC. The schools served primarily low-income students. All participants were enrolled in a CS: Principles course that was taught by an experienced CS teacher. The teachers ranged in teaching experience between 6–10 years, with three or more years of teaching CS. The CS: Principles course is an Advanced Placement (AP) course focused on computing in relation to societal needs, and the development of computational thinking skills. Teachers select the specific programming languages their students will use. The curricular framework for AP:CSP courses emphasizes abstraction, creativity in computing, problem analysis, communication, and collaboration. The framework explicitly addresses the following CT practices: abstraction, sequencing, iteration, testing and debugging. The course culminates in an AP test.

Eleven students participated in think alouds. All students responded to the same SRL prompts for forethought and reflection (described below), but not all students responded to the same CT problem in the performance phase, due to differences in teacher pacing through the CSP curricular framework. The four students at the charter school responded to a task henceforth referred to as the Fish Game; the other seven students responded to a task henceforth referred to as the Clicker Game. The two tasks were exactly alike in the SRL forethought and reflection prompts. They were alike in calling for the same CT algorithmic concepts, but differed in difficulty; for instance, where the Clicker Game required students to define and call only one function, the Fish Game required two functions. The Fish Game task work was treated as pilot material for performance scoring and coding of think alouds; only Clicker Game data, which were analyzed after protocols had been developed, are reported here.

2.3. Instruments

The FA task consists of three parts: a series of forethought questions, a CT problem to solve, and a series of reflection questions. These parts align with the phases of SRL. The forethought and reflection questions are adapted from Cleary, Callan, Malatesta, and Adams (2015), Cleary and Chen (2009), and Cleary and Zimmerman (2004) studies.

2.3.1. Forethought Questions

The task begins when students preview the CT problem they will attempt to solve and respond to 10 items intended to elicit forethought processes such as self-efficacy, task analysis, and pre-planning strategies. The self-efficacy question asks, "How confident are you that you can complete this task?" on a scale ranging from 1 (completely unconfident) to 6 (completely confident). The remaining items elicit students' task analysis processes and strategic planning through open-ended questions.

2.3.2. Computational Thinking Performance Problem

Students next proceed to the computer interface to attempt to solve the CT problem: the Fish Game or the Clicker Game. Both problems were adapted from learning units in the AP:CSP curriculum CS Principles (Code.org). The problems draw on the following CT algorithmic concepts: operators, events, sequencing, data, functions and conditionals. These concepts align with those listed by Brennan and Resnick (2012), except in omitting loops and adding functions.

2.3.3. Reflection Questions

The final part of the task consists of 12 items to elicit participants' self-reflection processes such as satisfaction of their own performance, reflection on CT understanding and strategy use, and adjustment. The satisfaction question asks, "How satisfied are you with the final product of your work on the computing task?" on a scale ranging from 1 (very unsatisfied) to 6 (very satisfied). The remaining items elicit students' reflection on problem-solving strategies, and adaptive inferences.

2.4. Think-Aloud Procedures

The think alouds were conducted concurrently with task completion under conditions that minimized extraneous factors to the extent possible within the school setting (i.e., in a secluded space). Each think-aloud participant sat with one of the authors or a trained research assistant. The think alouds started with standardized instructions, describing the nature and purpose of the interview. When a student previewed the task in the forethought phase, the researcher encouraged the students to begin to verbalize. This provided a warm-up opportunity, and allowed researchers to assess, for instance, whether the student's level of vocalization was appropriate, and whether it was necessary to explain again the nature of the research. Participants were audio-recorded as they responded to the CS task. Researchers only spoke in order to remind students to continue verbalization, if they lapsed into silence. Screen captures and artifacts of code were collected at the end of task completion. Such methods align with research-based recommendations, and have been shown to be useful to maximize validity of verbal reports in problems of moderate cognitive complexity (Ericsson & Simon, 1993; Russo et al., 1989; Someren, Barnard, & Sandberg, 1994; Taylor & Dionne, 2000).

2.5. Analytic Methods

The team developed separate methods for scoring artifacts of student code and for coding think-aloud data. Four members of the design team (the first author, third, and fourth authors, and another CS teacher) began the data analysis by constructing a scoring rubric for quantification of the artifacts of code. Prior to looking at any data, we drafted a rubric with three points (0 through 2) per dimension and the six types of CT algorithmic *concepts*: conditional use, data, operators, user-defined functions, events, and sequencing. The team members independently piloted scoring of students' coding artifacts from Fish Game work. After piloting the rubric, the group discussed adding an extra level of quality to resolve rater disagreement at the high end of the scale. We consulted a computer scientist, who supported the 4-point scale, and additionally suggested a seventh dimension for holistic appraisal of overall performance. We independently re-scored the same sample of Fish Game performance artifacts using the modified rubric, with the result of improved agreement among raters. The final rubric assessed seven areas of CT concepts with four levels of quality each, scored 0–3, resulting in 21 available points.

Our next step was to score the seven Clicker Game artifacts of computing code, and assess interrater reliability. Three raters independently scored each artifact. Over the seven criteria on which the artifacts were rated, each on a scale of 0 to 3 points, the percent of agreement within one point for any two pairs of raters ranged from 90.5% to 100%. The percent of absolute agreement ranged from 62% to 76%. The correlation between total scores (summed scores on all seven dimensions) awarded by any two pairs of raters ranged from .89 to .96.

To analyze the think-aloud data, we drafted a protocol with a priori codes and definitions based on the literature on SRL and CT. The a priori code list included four CT practices (*testing and debugging, iteration, abstraction, generalization*). It included multiple subprocesses in each phase of the SRL model. In the forethought phase, we coded for six subprocesses; in the performance phase, five subprocesses; in the reflection phase, four subprocesses. We tried out the coding protocol using multiple Fish Game think-aloud transcripts. The team began with each individual member highlighting what they conceived to be evidence of self-regulation or a computational thinking practice on the physical transcripts, with reference to the a priori definitions. A discussion of the highlighted areas began the norming process. To address any confusion among the raters about definitions of CT or SRL, we consulted experts in CS and SRL. Table 1 lists the targeted CT practices and SRL subprocesses, with coding guidelines. During norming, the team also added exemplary quotes to the coding protocol.

Table 1. Think-aloud coding protocol.

SRL Phase	Sub-process	Questions the process targets	Example transcripts
Elements of Self-Regulated Learning			
Forethought	Goal Setting	• Do you have a goal when you study for this task? Explain. • What goal are you trying to achieve on this task?	*I want to get an A on this.*
	Task analysis	• What concepts or skills do you know that will help you solve this task?	*I think I'm supposed to create a function to do this.*
	Strategic planning	• What strategies do you think you will use when preparing to complete this task?	*I'm going to use resources like the answers to the last quiz.*
	Self-efficacy	• How sure are you that you can solve 70% of these problems?	*I'm pretty sure I can complete this task (because . . .)*
	Intrinsic interest	• How much do you enjoy studying/preparing for computer science assessments?	*I've been practicing this because it's fun for me.*
	Goal orientation	• Why is it important for you to succeed on this task?	*I just don't want to do worse than the other people in my class.*
Performance	Attention focusing	• Do you try to motivate yourself when working? • What do you do when you don't feel like working on this task?	*Ok, I have to concentrate now.*
	Self-instruction /imagery	• Do you quiz or ask yourself questions during your work? • What diagrams/drawings/graphs do you use to help you understand the concepts?	*[Not found in transcripts]*
	Help seeking	• What do you do when you need help with understanding a concept or solving a problem?	*There's a lesson that we did that I can look at that does the same thing.*
	Self-recording	• Do you keep track of where or how you work? • Do you keep track of how long you work?	*I like to write in comments because it helps me keep track of what I'm doing.*
	Metacognitive monitoring	• Do you check your solutions to see if they give correct results?	*OK, I'm going to check this and see if it runs.*
Self-reflection	Self-evaluation	• What grade would you need to feel completely satisfied?	*I think I completed this pretty well.*
	Attribution	• What is the main reason you did well (or poorly) on this task?	*I couldn't do that part because I never really learned it, I wasn't paying enough attention*
	Self-satisfaction	• How satisfied are you with your performance? • How satisfied are you with this achievement level?	*I feel like I accomplished something.*
	Adaptive inferences	• What do you need to do to improve your performance on for next time? • What do you need to do to perform well on your next assessment?	*If I did this again, I'd first review the lesson instead of wasting a lot of time trying random things.*

CT Practice	Description
Elements of Computational Thinking (based on Brennan & Resnick, 2012)	
Testing & debugging	Students develop habits of running programs and sub-programs to test, and problem-solve based on error analysis and prior learning. – for this to count in interpretation of transcripts, students must test their own code, not just run code from a resource. There are different types of debugging, including trial and error, and reading and interpreting error messages.
Iteration	Students use both reiterated algorithms to perform similar tasks. Continuing use of algorithms or structures to perform similar tasks that build incrementally to a goal. Compare to reusing and remixing which is adopting other work and/or modifying code from an external source.
Abstraction & modularizing	Students identify patterns of key structural similarities of different tasks so as to be able to solve multiple problems given different variables. Top down design/ Breaking down a complex task into chunks.
Generalization: Reusing & remixing	Students refer to work of their own, of peers, or other resources to generate new solutions to similarly structured problems. Reusing and remixing is intentional use of an outside source that can be adopted or modified. To differentiate it from help seeking, reusing and remixing requires knowledge of the structure of the problem that indicates a specific resource can be adopted or modified for a solution. Help seeking, on the other hand, occurs when the student lacks such knowledge.

The full protocol has been slightly simplified for publication purposes.

After development of and practice with the think-aloud coding protocol, we proceeded to code the seven Clicker Game tasks. The first author assigned transcripts to two raters per transcript, who worked independently to code the think alouds according to the annotated code definitions. After two raters had coded each transcript, the team met to reach consensus on the analysis in areas one might have missed or other discrepancies in coding.

3. Results

Scores on the artifacts of computer code from the Clicker Game task (n = 7) were calculated by summing scores over the seven CT concepts, averaging across raters, and rounding to the nearest whole number. The observed scores ranged from 4 to 21, out of 21 possible points. The average score was 15 (SD = 5), or 72%. The average length of processing time was 41 minutes (SD = 12), which is a long time for a classroom assessment. Thinking aloud tends to lengthen performance time (Russo et al., 1989). Under regular classroom conditions, the FA task could likely be completed within a single class session. The average length of written lines in a transcript was 134 (SD = 51).

To communicate the evidence from the think alouds, we report the results of our analysis graphically for four students (Figures 1 and 2). We select only four students for simplicity of presentation. Of the nonselected three, one performed very poorly and their cognitive processes consisted largely of seeking help by using resources indiscriminately, i.e., scrolling through all lessons offered during the school year rather than ones directed at concepts relevant to the Clicker Game. This student also engaged in negative self-attributions frequently during performance and reflection.

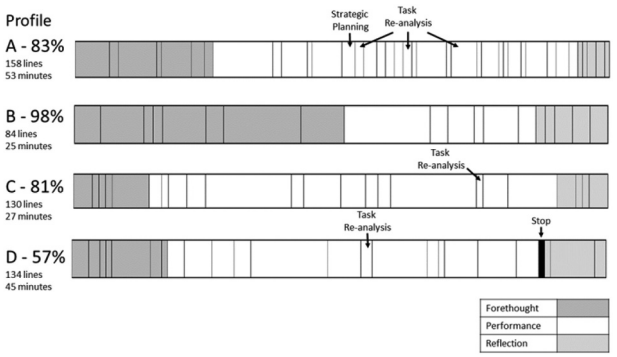

Figure 1. Relative proportions of verbalizing by students in three phases of FA task.

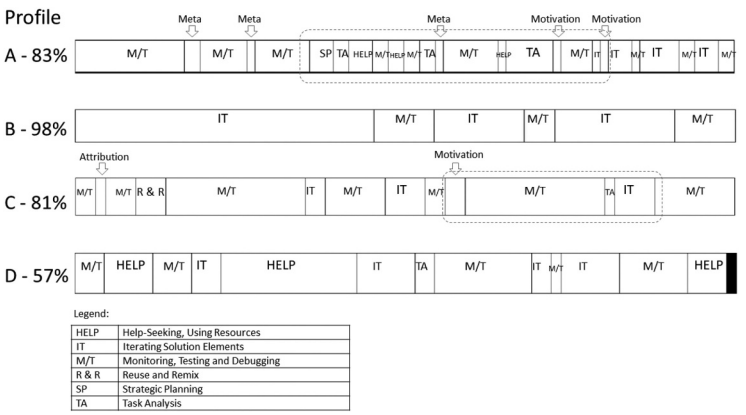

Figure 2. Detail of cognitive and metacognitive processes at performance phase on FA task.

Another student's results are not displayed because of strong similarities with other profiles (Profiles A and C), both in performance score and use of CT and SRL. The last student's results are not displayed due to general lack of SRL; the student scored 90% and either did not use or lacked cognitive access to their self-regulative processes.

To construct the profiles, we used lines of transcript as a pseudo-measure for amount of cognitive processing (rather than processing speed). We then created linear profiles of student thinking during task completion. The lengths of the bars were standardized to help the reader avoid focusing on time and verbosity, which are not the main interest of this study; these are, however, displayed next to each score. For each individual profile, the width of a segment on the bar represents the *relative* amount of processing of a given type process, in the context of the amount of other types of processes used by the same student.

3.1. Research Question 1: Do Student Think-alouds Provide Evidence that Computing Problems in the FA Task Elicit Cognitive Processes Consistent with Conceptual Definitions of CT?

Figure 2 depicts the same illustrative cases as previously shown. We found that in the performance phase of the FA task, student cognitive processes were related to some aspects of CT, but not all were represented equally extensively. Students used testing and debugging and iterations frequently. They were found to use abstraction only on three occasions, two of which were found in Profile B, the highest-performing student. The following statement gives an example of this student's ability to recognize patterns underlying the problem's surface features: "Because with almost every game that I've made on Code Studio [the programming environment] pretty much always have a variable that increased and decreased accordingly, depending on an action." A different type of abstraction was identified when a student, not depicted here, said, "I already see the code in my head." Both these statements occurred during forethought. CT generalization (reusing and remixing) was observed with the lowest frequency, only twice, both times during the performance phase. The student in Profile C showed reusing and remixing by stating, "Where I can find something similar to see what I code. . . . Oh actually, I can use my app that I created." Another student, not depicted here, also had the idea of reusing and remixing: "What did I name this before? And how I could compare it to be like a replica of what I'm trying to do right now."

3.2. Research Question 2: Do Student Think-alouds Provide Evidence that SRL Prompts Embedded in the FA Task Elicit Cognitive Processes Consistent with SRL Theory?

Analysis of the bars in Figure 1, read from left to right, shows that students engaged in substantial amounts of forethought and reflection. They also engaged in processes associated with forethought during the performance phase, pointed out in the figure with arrows. Looking within the forethought phase at specific codes, we found that most of the SRL prompts elicited thinking that corresponded with theoretical descriptions of forethought subprocesses, including task analysis, estimation of task self-efficacy, and strategic planning. We found, however, that several students had difficulty responding to items intended to call upon task analysis, and did not use the forethought phase to focus on specific concepts on which they were being formatively assessment (e.g., loops, conditionals). For instance, in response to a forethought item prompt intended to prompt thinking about CT concepts, one student stated they would "try to see every aspect of it, be open minded." However, this was the lowest-scoring student; it is possible that they understood the question but had at their mental disposal few or no CT concepts with which to respond. While students spent different amounts of processing time in forethought, no students avoided forethought or were miscued by the directions to begin problem solving when expected to be analyzing or planning. One student engaged in abstraction (a CT practice) before completing the forethought items. Very few task-irrelevant mental processes were found in the transcripts; in a few cases

students uttered phrases indicating distraction by noise, or needed to do something like find a pencil.

During the performance phase, there was more of a mixture and melding of SRL and CT processes, which is discussed below as it relates to the fourth research question. With regard to SRL alone, in the performance phase students used self-monitoring extensively, and the majority did some help seeking by referring to online lessons and projects completed in their class. Six of the seven students returned to the forethought phase in the SRL model for task re-analysis and additional strategizing, as shown with arrows in Profiles A, C, and D, Figure 1.

Most students spent about 10% to 15% of their time in the reflection phase (Figure 1), although one mid-performing student and the lowest-scoring student had very brief reflections. The reflection phase was similar to forethought in that students almost entirely focused on SRL, as task items prompted them to do. No trends appeared in ratings of self-satisfaction. Some students expressed positive self-satisfaction, and others were self-critical; these judgments seemed to be unrelated to performance.

Some students made plans to adapt their learning during reflection, but others among the seven declared they would not change or adapt any of the strategies they used in the future, even though they were not fully satisfied with their performance. A student with about average performance said, "in the future, I would approach it the same way, just because I would understand it like that, and I wouldn't do anything differently." Three students made attributions for their performance, all involving practice or the lack thereof. The highest-performing student attributed understanding to effort: "practice doing them over and over again though, so that I would understand a little bit more and see the pattern." When students attribute their success or failure to internal, controllable, and unstable factors such as effort, they are more likely to be motivated to work hard to improve future performance (Weiner, 2010). The statement quoted above not only indicates a positively motivating attribution, but, apparently, intrinsic motivation to set a goal of CT abstraction (pattern recognition).

3.3. Research Question 3: How Do Students at Different Levels of Mastery Vary in Uses of CT and SRL?

We compared the students' scores for the artifacts of code they had produced during the FA task with their cognitive and metacognitive processes shown in the think alouds. The artifacts of code had been scored for the algorithmic conceptual component of CT. Profiles B and D varied both in scores (98% versus 57%), and the processes displayed in their think alouds. They varied in the relative amount of cognitive processing they used in different phases of SRL, particularly in their forethought and performance phases (Figure 1). Relative to their total time on the FA task, the student in Profile B, who scored 98% on the task, spent the most time in forethought. This student devoted approximately half of their cognitive time to task analysis and planning, and took relatively little time for actual performance. Profile D, on the other hand, who scored 57%, spent proportionately more time in performance than in the other two phases. This student had to stop working on a solution due to time constraints in their classroom, as shown with a black bar on the graphic near the right-hand end. Had they not been forced to stop, their performance phase would have been longer.

Figure 2 shows each student's mental processes during the performance phase. When Profile B began their solution attempt on the Clicker Game, they set to work employing CT concepts immediately, using the CT practices of performing strategies iteratively with minor variations, to build in increments toward the solution. They never sought help in any form. Rather than using the trial and error approach that required frequent testing and debugging, the student worked iteratively toward a partial solution, then performed purposeful testing to monitor their progress. At the end of performance, they self-evaluated by comparing their solution to the task goals: "just double checking the work."

Profile D shows a very different approach to problem solving in the performance phase, and one that sets them apart from all six other students, including the student with the lowest score. Starting from the point where the student entered the performance phase of the task, 20 of 74 lines from their

transcript (27%) were coded as help seeking, using resources such as prior lessons or completed work of their own. Although five out of seven students used external resources, they did so in briefer bursts. Profile D used resources heavily up until nearly the halfway point in their processing time, and had returned to seeking help when their time ran out.

The two remaining profiles in Figure 2, A and C, represent the thinking of students who scored in the middle range on the FA task. The students differed in some respects. Profile A went to resources frequently, while Profile C only referred to resources once, and then on purpose to reuse and remix. However, both students, as well as a third student not depicted here, demonstrated a similar cognitive pattern during their performance phase. First, they used trial and error often and at length. The raters classified such trial and error processes both as monitoring (SRL) and testing and debugging (CT). Then, the students at some point returned to task analysis and/or planning, processes characteristic of forethought, engaging in a cognitive and metacognitive back-and-forth wherein they re-considered task parameters, tried out strategies, obtained feedback from trials, and debugged. In both cases, though to different extents, motivation and return to task analysis preceded the students' ability to make incremental gains through CT iteration. Figure 2 highlights such sequences of cognitive and metacognitive processes with dashed-line boxes. Reading Profile A from left to right, the student failed to make progress through iterative problem solving until nearly three-quarters of the way through their performance phase. However, after several returns for re-planning and reexamination of the task, the student was eventually able generate solutions to problem features and use them iteratively to task completion. Both Profile A and C also used self-motivation, for example: "Every time I come across a stopping point, I always find a way to do it."

Profile A, C, as well as the profile of a third mid-performing student not depicted here used unsophisticated trial-and-error approaches to problem solving with frequent monitoring through testing and debugging, followed by reapplication of SRL forethought processes to achieve incremental gains. As rated on the scoring rubric for their artifact of code, all three of the students met basic specifications in their Clicker Game solution. However, none used functions, a targeted CT concept which would have made their solutions more efficient. Thus, the performance scores and evidence from the think alouds show agreement: both sources suggest that these students had in common a working but inefficient approach to computing. This congruence between performance evidence on the CT task and evidence of student mental processes supports construct validity; variance in performance scores could be clearly associated with variability in cognitive processes.

3.4. Research Question 4: Do Think Alouds Provide Evidence that Elucidates Definitions of the CT Construct or SRL Processes?

The analysis of the think alouds showed that some CT practices were highly similar to subprocesses articulated in the Zimmerman (2000) SRL model. This was particularly true in the case of testing and debugging (CT) and metacognitive monitoring (SRL). In CT, testing and debugging has been defined as a practice to "detect and identify errors, and then fix the errors, when a solution does not work as it should" (Shute et al., 2017, p. 12). In SRL, Zimmerman (2013) describes metacognitive monitoring as "informal mental tracking of one's performance processes and outcomes" (p. 137); in a computing environment, this would occur when a student tracked their outcome by testing and mentally reconsidered their process to debug. Testing and debugging was exhibited very frequently in the transcripts during the performance phase, and was almost always interpreted by the raters as concurrent with metacognitive modeling. Figure 2 therefore uses a single code (Monitoring and Testing/Debugging, M/T) to represent this process that was shared by CT and SRL. The very few instances when metacognitive monitoring *without* testing and debugging was identified are represented in Figure 2 by the letters "Meta" over the corresponding segment of transcript. As an instance of this type of metacognitive monitoring, the student in Profile A stated, "I'll come back to this later, I don't want to spend too much time on one thing."

In analyzing the transcripts, we also found that the definition we began with for SRL help seeking made it hard to differentiate that process from CT generalization (reusing and remixing) – both involved accessing an external resource. However, raters were able to clarify the distinction between SRL help seeking and CT generalization fairly readily through close examination of the transcripts and reference to the CT literature. We added the specification to the coding protocol that accessing a resource in generalization involved using the resource to select elements of code that the user knew to contain *general* structural features that would help solve a *specific* problem (Table 1). Help seeking was interpreted as a more exploratory process. Thus, examination of the data provided a more refined understanding of contrast between generalization and SRL help seeking, at least as it manifested itself in student thinking during performance of this FA task.

4. Discussion

This study revealed that tasks designed to assess computational thinking with embedded self-regulated learning prompts elicited cognitive processes aligned with theoretical definitions of CT and SRL. Among CT practices, testing and debugging and iteration predominated. More limited abstraction and generalization were found. Among the SRL subprocesses, students engaged most extensively in task analysis (or re-analysis), monitoring, and help seeking. Some students made attributions and adaptive inferences. Neither self-recording nor self-instruction was found. The few observed processes not directly related to CT or SRL were primarily motivational or otherwise metacognitive, and not distracting or irrelevant to performance. These findings provided evidence of the construct validity of interpretations derived from the tasks.

Methodologically, we found little evidence of reactivity, although the task took longer to complete than anticipated. The combination of think-alouds with artifacts of code yielded sufficient information for analysis of the relationship between processes and performance. The graphical display of results proved useful for depicting and interpreting not only type of cognitive process, but duration and sequence. Stacking graphics across different profiles provided a holistic comparison of how different students engaged in different processes when solving the task.

We compared cognitive and metacognitive processes from the think alouds with rubric-based scores for the artifacts of code the students produced during the FA task. Cognitive and metacognitive processes were used quite differently by students at different levels of mastery. The cognitive processes associated most clearly with high performance were iteration (CT), and task analysis and planning (SRL). When the highest-performing student entered the performance phase of the FA task, they used SRL practices very little, except for a few strategic instances of testing and debugging (also coded as monitoring). Instead, the student engaged mostly in the cognitive process of iterating CT algorithmic concepts in ways that built incrementally to a successful solution. This result converged with empirical findings of Zimmerman and Kitsantas (1999), who found that when students approached mastery in performance, their attention shifted from processes to outcome goals. Only the higher-performing students showed CT generalization or abstraction.

Quantity, placement in sequence, and quality of task analysis all appeared to differentiate students at different levels of mastery. The highest-performing student spent a much greater proportion of their cognitive processing time in task analysis than did their lower-performing peers. All of their task analysis time occurred in the forethought phase. In terms of quality, we found that the lowest-performing students had difficulty even listing CT algorithmic concepts they had learned, which they might use in their solution. Students at moderate performance levels were able to list several concepts they might use. At the upper end of the performance spectrum, we found students whose task analysis included recognition of structural patterns associated with CT abstraction.

The think-aloud data shed light on definitional questions about the CT construct. CT testing and debugging was often indistinguishable to our raters from SRL monitoring. Students exhibited both as repeated goal-directed thinking to check a solution or steps in a solution during performance. This

suggests that testing and debugging may be little different from self-monitoring during problem solving, applied to computing environments. The specifically computational nature of testing and debugging requires elaboration. There is continued debate in the CS-education field as to whether CT is a construct in its own right, or a part of algorithmic thinking that is common to multiple domains, such as mathematics (Shute et al., 2017). The argument against CT as "universally applicable" (Wing, 2006) would be supported if further research indicates that testing and debugging are not CT-specific ways of thinking, but domain-specific applications of monitoring.

Regarding other SRL processes, we found that the extent, type, and placement of help seeking were indicative of student learning needs. The students who resorted to external resources early and for the longest time (Student D and another student, not displayed) both scored at the low end on the performance rubric. These students spent time consulting resources they selected apparently at random. Students who were more proficient used monitoring to identify their task-specific needs; they resorted to external resources only if multiple repetitions of trial and error had not helped them find a solution. When they sought help, they purposefully searched for relevant online lessons or their own prior coding. This empirical observation is consistent with prior research on differentiated use of help and resources among students with varying competence and sophistication in SRL (Karabenick & Gonida, 2018).

We confirmed other empirical findings that students could compensate for a "weak start" by returning to the SRL forethought processes of repeated task analysis and strategic planning (Karabenick & Gonida, 2018). The return to task analysis and planning after entry into the performance phase provides a cautionary reminder that Zimmerman's (2000) phase model of SRL should not be oversimplified to the idea that SRL is simply a series of processes. As Usher and Schunk (2018) state, SRL subprocesses can occur at any phase, given the complexity of human functioning. This may have been particularly true with problems such those in the FA task we studied, which was moderately challenging.

4.1. Limitations

We recognize several limitations to our work. The first is the sample size, which prohibits quantitative analyses and limits generalizability. The amount of time required to analyze think aloud protocols makes it challenging to conduct large-scale research. We are therefore seeking ways to automate at least some aspects of coding through natural-language processing (Magliano & Graesser, 2012). We also bear in mind that there are multiple operational definitions of CT and multiple theories of SRL. In our analyses we applied particular models of CT (Brennan & Resnick, 2012; Shute et al., 2017) and SRL (Zimmerman, 2000) that informed our design and analyses. Application of different models of either CT or SRL to our task design and analysis likely would have yielded different results. A third limitation is that, like other research-based instruments to measure CT (Grover et al., 2015; Moreno-León & Robles, 2015; Werner et al., 2012), our task was embedded in a specific programming environment (Code Studio). Our continuing work on task development involves offline performance measures that are appropriate to multiple courses addressing the same CT concepts and practices.

4.2. Implications for Further Research and Development

Further research is needed to investigate cognitive processes of more expert CT learners, to determine whether the limited use of CT practices such as abstraction and generalization found in this study is associated primarily with students at a novice level. Further research is also warranted to micro-analyze the subprocess of help seeking in SRL. The types of resources and the way students used them varied greatly, and in one case use of resources was, in our estimation, an instance of CT reusing and remixing, rather than SRL help seeking. Future research should therefore examine help seeking in depth.

Regarding implications for further development of these tasks and SRL-based FA tasks that involve similar types of problem solving, we recommend work on feasible ways for teachers to collect physical traces of problem-solving practices and SRL cognitive and metacognitive processes. In this study, frequent CT testing and debugging and SRL help seeking seemed to be associated with relatively poor CT performance. Therefore, we have been working to develop ways for students to self-record CT practices such as testing and debugging through using comment features in code. This would allow teachers to better monitor the use of help seeking and make instructional adjustments accordingly. Also, assessment developers should design classroom-friendly ways to measure subprocesses such as task analysis, so that teachers and students can track progress in SRL over time. Incorporation of directions that allow students to select the resources they wish to use, as suggested by Karabenick and Gonida (2018), might also help students and teachers track changes in need for academic scaffolding among their students.

5. Conclusion

The overall design approach for the formative assessment tasks examined in this study rested on three assumptions, for each of which we needed to provide warrants through validation. First, we needed to provide evidence for the claim that teachers could interpret student performance on the formative assessment task in terms of computational thinking. Second, we needed to provide evidence that task prompts designed to elicit self-regulated learning did so. Without engagement and practice in self-regulated learning, students are unlikely to reap its learning benefits. The results of this study demonstrated warrants for these two assumptions. Evidence from think alouds showed that performance on the task involved computational and self-regulated learning cognitive and metacognitive processes, with no significant irrelevant sources of variance.

A third assumption underlies this entire design approach to formative assessment of computational thinking: overlapping mechanisms between formative assessment and self-regulated learning can be leveraged to boost the potential of either one, taken separately, to support learning in computational thinking. Whether classroom use of a formative assessment task such as the one described in this study can result in student learning gains can only be revealed through experimental or quasi-experimental methods. Researchers in assessment who support the idea that formative assessment activates and thereby promotes student learning should be prepared to use such rigorous methods to argue their claims.

Disclosure Statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Sarah M. Bonner, Educational Foundations and Counseling Programs, Hunter College, City University of New York, New York, USA.

Peggy P. Chen, Educational Foundations and Counseling Programs, Hunter College, City University of New York, New York, USA.

Kristi E. Jones, New York City Department of Education, New York, USA.

Brandon A. Milonovich, Ardley Union Free School District, Ardsley, USA.

References

Aho, A. V. (2011). Computation and computational thinking. *Computer Journal*, *55*(7), 832–835. doi:10.1093/comjnl/bxs074

Andrade, H., & Brookhart, S. M. (2016). The role of classroom assessment in supporting self-regulated learning. In D. Laveault & L. Allal (Eds.), *Assessment for learning: Meeting the challenge of implementation* (pp. 293–309). Springer, Cham: Springer International Publishing.

Bennett, R. E. (2011). Formative assessment: A critical review. *Assessment in Education: Principles, Policy & Practice*, *18* (1), 5–25.

Black, P., & Wiliam, D. (1998). Assessment and classroom learning. *Assessment in Education: Principles, Policy & Practice*, *5*(1), 7–74.

Bonner, S. M. (2013). Validity in classroom assessment: Purposes, properties, and principles. In *Sage Handbook of Research on Classroom Assessment* (ed., J. H. McMillan, pp. 87–106). Sage Publications: Thousand Oaks, CA doi:10.4135/9781452218649

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. *Paper presented at the annual meeting of the American Educational Research Association*, Vancouver, Canada. Retrieved from http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Broadfoot, P. M., Daugherty, R., Gardner, J., Harlen, W., James, M., & Stobart, G. (2002). *Assessment for learning: 10 principles*. Cambridge, UK: University of Cambridge School of Education.

Callan, G. L., & Cleary, T. J. (2019). Examining cyclical phase relations and predictive influences of self-regulated learning processes on mathematics task performance. *Metacognition and Learning*, *14*(1), 43–63. doi:10.1007/s11409-019-09191-x

Chen, P. P., & Bonner, S. M. (2020). A framework for classroom assessment, learning, and self-regulation. Assessment in Education: Principles, *Policy & Practice*, *27*(4), 373–393 doi:10.1080/0969594X.2019.1619515

Cleary, T. J. & Chen, P. P. (2009). Self-regulation, motivation, and math achievement in middle school: Variations across grade level and math context. *Journal of School Psychology, 47*, 291–314. doi:10.1016/j.jsp.2009.04.002

Cleary, T. J., Callan, G. L., Malatesta, J., & Adams, T. (2015). Examining the level of convergence among self-regulated learning microanalytic processes, achievement and a self-report questionnaire. *Journal of Psychoeducational Assessment*, *33*(5), 439–450. doi:10.1177/0734282915594739

Cleary, T. J., & Zimmerman, B. J. (2004). Self-regulation empowerment program: A school-based program to enhance self-regulated and self-motivated cycles of student learning. *Psychology in the Schools*, *41*(5), 537–550. doi:10.1002/pits.10177

Computer Science Teacher Association. (2017). *CS standards*. CSTA K-12 Computer Science Standards. Retrieved from https://www.csteachers.org/page/standards

Cronbach, L. J. (1971). Test validation. In R. L. Thorndike (Ed.), *Educational measurement* (2nd ed., pp. 443–507). Washington, DC: American Council on Education.

Department of Labor Bureau of Labor Statistics. (2020). *Occupational outlook handbook*. https://www.bls.gov/ooh/home.htm

DiBenedetto, M. K., & Zimmerman, B. J. (2010). Differences in self-regulatory processes among students studying science: A microanalytic investigation. *International Journal of Educational and Psychological Assessment*, *5*(1), 2–24.

DiGiacomo, G. & Chen, P. P (2016). Enhancing self-regulatory skills through an intervention embedded in a middle school mathematics curriculum. *Psychology in the Schools*, *53*(6), 601–616. doi:10.1002/pits.2016.53.issue-6 6

Downing, S. M. (2006). Twelve steps for effective test development. In S. M. Downing & T. M. Haladyna (Eds.), *Handbook of test development* (pp. 3–26). Mahwah, NJ: Lawrence Erlbaum Associates.

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

Fox, M. C., Ericsson, K. A., & Best, R. (2011). Do procedures for verbal reporting of thinking have to be reactive? A meta-analysis and recommendations for best reporting methods. *Psychological Bulletin*, *137*(2), 316–344. doi:10.1037/a0021663

Grover, S., & Basu, S. (2017). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education (pp. 267–272).

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43. doi:10.3102/0013189X12463051

Karabenick, S. A., & Gonida, E. N. (2018). Academic help seeking as a self-regulated learning strategy. In D. H. Schunk & J. A. Greene (Eds.), *Handbook of self-regulation of learning and performance* (2nd ed., pp. 421–433). New York, NY: Routledge.

Kelley, T. R., Capobianco, B. M., & Kaluf, K. J. (2015). Concurrent think-aloud protocols to assess elementary design students. *International Journal of Technology and Design Education*, *25*(4), 521–540. doi:10.1007/s10798-014-9291-y

Kingston, N., & Nash, B. (2011). Formative assessment: A meta-analysis and a call for research. *Educational Measurement: Issues and Practice*, *30*(4), 28–37. doi:10.1111/j.1745-3992.2011.00220.x

Leighton, J. P. (2004). Avoiding misconception, misuse, and missed opportunities: The collection of verbal reports in educational achievement testing. *Educational Measurement: Issues and Practice*, *23*(4), 6–15. doi:10.1111/j.1745-992.2004.tb00164.x

Leighton, J. P. (2017). *Using think-aloud interviews and cognitive labs in educational research*. New York, NY: Oxford University Press.

Magliano, J. P., & Graesser, A. C. (2012). Computer-based assessment of student-constructed responses. *Behavior Research Methods*, *44*(3), 608–621. doi:10.3758/s13428-012-0211-3

Mason, L. H. (2004). Explicit self-regulated strategy development versus reciprocal questioning: Effects on expository reading comprehension among struggling readers. *Journal of Educational Psychology*, *96*(2), 283–296. doi:10.1037/0022-0663.96.2.283

Messick, S. (1989). Validity. In R. L. Linn (Ed.), *Educational measurement* (3rd ed., pp. 13–103). New York, American Council on Education: Macmillan.

Moreno-León, J., & Robles, G. (2015). Dr. Scratch: A web tool to automatically evaluate Scratch projects. In *Proceedings of the workshop in primary and secondary computing education* (pp. 132–133). London, UK.

New York City Department of Education. (2019, December 12). *Chancellor Carranza announces 72 percent increase in students taking computer science since launch of computer science for all*. https://www.schools.nyc.gov/about-us/news

Nisbett, R. E., & Wilson, T. D. (1977). Telling more than we can know: Verbal reports on mental processes. *Psychological Review*, *84*(3), 231–259. doi:10.1037/0033-295X.84.3.231

Paris, S. G., & Paris, A. H. (2001). Classroom applications of research on self-regulated learning. *Educational Psychologist*, *36*(2), 89–101. doi:10.1207/S15326985EP3602_4

Pellegrino, J. W., Chudowsky, N., & Glaser, R. (Eds). (2001). *Knowing what students know: The science and design of educational assessment*. Washington, DC: National Academy Press.

Pepper, D., Hodgen, J., Lamesoo, K., Kõiv, P., & Tolboom, J. (2018). Think aloud: Using cognitive interviewing to validate the PISA assessment of student self-efficacy in mathematics. *International Journal of Research & Method in Education*, *41*(1), 3–16. doi:10.1080/1743727X.2016.1238891

Peters-Burton, E. E., Cleary, T. J., & Kitsantas, A. (2015, October). Computational thinking in the context of science and engineering practices: A self-regulated learning approach. *Paper presented at the 12th international conference on cognition and exploratory learning in digital age*, Dublin, Ireland.

Román-González, M., Moreno-León, J., & Robles, G. (2017). Complementary tools for computational thinking assessment. In S. C. Kong, J. Sheldon, & K. Y. Li (Eds.), *Proceedings of international conference on computational thinking education (CTE 2017)* (pp. 154–159). Hong Kong, China: The Education University of Hong Kong.

Román-González, M., Perez-González, J. C., & Jimenez-Fernandez, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behavior*, *72*, 678–691. doi:10.1016/j.chb.2016.08.047

Russo, J. E., Johnson, E. J., & Stephens, D. L. (1989). The validity of verbal protocols. *Memory & Cognition*, *17*(6), 759–769. doi:10.3758/BF03202637

Shepard, L. A., Penuel, W. R., & Pellegrino, J. W. (2018). Using learning and motivation theories to coherently link formative assessment, grading practices, and large-scale assessment. *Educational Measurement: Issues and Practice*, *37*(1), 21–34.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158. doi:10.1016/j.edurev.2017.09.003

Siddiq, F., & Scherer, R. (2017). Revealing the processes of students' interaction with a novel collaborative problem solving task: An in-depth analysis of think-aloud protocols. *Computers in Human Behavior*, *76*, 509–525. doi:10.1016/j.chb.2017.08.007

Someren, M. W. V., Barnard, Y. F., & Sandberg, J. (1994). *The think aloud method: A practical guide to modelling cognitive processes*. London, UK: Academic Press.

Taylor, K. L., & Dionne, J. P. (2000). Accessing problem-solving strategy knowledge: The complementary use of concurrent verbal protocols and retrospective debriefing. *Journal of Educational Psychology*, *92*(3), 413. doi:10.1037/0022-0663.92.3.413

Usher, E. L., & Schunk, D. H. (2018). Social cognitive theoretical perspective of self-regulation. In D. H. Schunk & J. A. Greene (Eds.), *Handbook of self-regulation of learning and performance* (2nd ed., pp. 19–35). New York, NY: Routledge.

Weiner. (2010). The development of an attribution-based theory of motivation: A history of ideas. *Educational Psychologist*, *45*(1), 28–36. doi:10.1080/00461520903433596

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on computer science education* (pp. 215–220). New York, NY: ACM.

Wiliam, D., & Thompson, M. (2008). Integrating assessment with learning: What will it take to make it work? In C. A. Dwyer (Ed.), *The future of assessment: Shaping teaching and learning* (pp. 53–82). Mahwah, NJ: Lawrence Erlbaum Associates.

Wilson, T. D. (1994). The proper protocol: Validity and completeness of verbal reports. *Psychological Science*, *5*(5), 249–252. doi:10.1111/j.1467-9280.1994.tb00621.x

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

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725.

Yadav, A., Burkhart, D., Moix, D., Snow, E., Bandaru, P., & Clayborn, L. (2015). *Sowing the seeds of assessment literacy in secondary computer science education: A landscape study*. Computer Science Teachers Association. https://csta.acm.org/Research/sub/Projects/AssessmentStudy2015.html

Zimmerman, B. J., & Kitsantas, A. (1999). Acquiring writing revision skill: Shifting from process to outcome self-regulatory goals.. *Journal of Educational Psychology, 91*2), 241-250.

Zimmerman, B. J. (2000). Self-efficacy: An essential motive to learn. *Contemporary Educational Psychology*, *25*(1), 82–91. doi:10.1006/ceps.1999.1016

Zimmerman, B. J. (2002). Becoming a self-regulated learner: An overview. *Theory into Practice*, *41*(2), 64–70. doi:10.1207/s15430421tip4102_2

Zimmerman, B. J. (2013). From cognitive modeling to self-regulation: A social cognitive career path. *Educational Psychologist*, *48*(3), 135–147. doi:10.1080/00461520.2013.794676

Zimmerman, B. J., & Schunk, D. H. (2011). *Handbook of self-regulation of learning and performance*. New York, NY: Routledge.

Zumbo, B. D., & Hubley, A. M. (Eds.). (2017). *Understanding and investigating response processes in validation research* (Vol. 26). Cham, Switzerland: Springer.