Robotics and Coding: A Framework for Examining Cognitive Demand

Anna Bloodworth, AnnaMarie Conner, Claire Miller, Lorraine Franco, Timothy Foutz, Roger B. Hill

Abstract

Drawing on research in mathematics and science education that has supported student higher-level thinking in K-12 classrooms, we sought to classify the ways of thinking and reasoning supported by robotics coding tasks. As part of a larger project, we examined coding tasks implemented in elementary school classrooms and analyzed ways of thinking and reasoning about coding supported by these tasks. From this analysis, we developed a framework for analyzing the cognitive demand of coding tasks that can be used to support researchers, curriculum developers, and teachers in supporting students' higher levels of thinking as they learn coding.

Keywords: coding, cognitive demand, framework, task

Bloodworth, A., Conner, A., Miller, C., Franco, L., Foutz, T., Hill, R. (2023). Robotics and Coding: A Framework for Examining Cognitive Demand, *35*(1), 7-31. https://doi.org/10.21061/jte.v35i1.a.1

Introduction

One of the features of Technology and Engineering Education that makes it so desirable within the school curriculum is the way it "promotes interdisciplinary connections with other school subjects" (International Technology and Engineering Educators Association [ITEEA], 2020, p. 9). In a similar manner, research involving faculty from multiple STEM content areas can provide insight into learning that can greatly enhance understanding of disciplinary content and practices. Thinking about enhancing such understanding in the context of coding, we bring together expertise from all four STEM education research areas to present a framework developed from a larger research project that examined what happens when we introduce coding through robotics activities in school settings.

Robotics is one of many technologies that technology and engineering educators have embraced over recent decades. It is part of the first context described in the Standards for Technological and Engineering Literacy (ITEEA, 2020) and has been introduced in classrooms from pre-Kindergarten through high school. The Standards for Technological and Engineering Literacy (STEL) link robotics with computation and computational thinking, so it is appropriate that this study was guided and influenced by partners in mathematics education.

In addition to being crucial in technology and engineering education, coding can be found more broadly in integrated STEM settings. Coding allows students to control the actions of robots to meet the challenges that might be presented within a Technology and Engineering Education Collegiate Association competition (International Technology and Engineering Educators Association, n.d.) or to accomplish goals set out in an Engineering byDesign (ITEEA, 2021) unit of instruction. Coding is also an activity that can be found in elementary technology and engineering education activities involving robotics. The usual form it takes there is block-based coding with tools like Scratch or variants[1] of that programming environment. Proponents of teaching computer science concepts, including coding, in K-12 schooling stress the importance of intentionally planned learning and have provided significant resources for elementary teacher professional development as well as elementary curriculum and instructional materials.

Coding activities help nurture creativity and problem-solving skills to prepare students for future careers (Chevalier et al., 2020; Yadav et al., 2016). Computing occupations make up 67% of all projected new jobs in STEM fields (U.S. Bureau of Labor Statistics, 2022); early access to coding can help prepare students for these kinds of careers. Innovation, consisting of creativity, problem-solving, higher-order thinking, entrepreneurship, and technological literacy, is one of three essential elements in preparing persons for successful work in twenty-first century occupations (Rojewski & Hill, 2014); coding and robotics activities provide an excellent platform for stimulating learning in this area. Programming, synonymous with coding, is mentioned several times in the STEL and is clearly a component that technology and engineering education professionals address within their instructional programs.

High-quality instruction can be characterized by high levels of engagement, alignment, rigor, and the degree to which the instruction is centered around what we, as educators, want students to know and be able to do (Bransford et al., 2002; Early et al., 2014; Weiss & Pasley, 2004). Relative to instruction involving robotics and coding, students should have opportunities to grapple with complex robotics coding activities in ways that encourage student engagement in a rigorous way that is aligned to academic standards. Building on work examining mental processes in technology education learning activities in the 1990's (Hill, 1997), Hill (2006) identified some key elements that would characterize the move to identify with engineering and engineering design as a focus for technology education. Analysis and optimization were being embraced at that time as technology educators moved away from trial-and-error approaches for hands-on laboratory activities. In the same way that incorporating engineering design brought new rigor to the field, giving attention to the cognitive demand, or intensity of intellectual work required, in coding

[1] Scratch (https://scratch.mit.edu), Ozblockly (https://ozobot.com/create/ozoblockly/), Rogic (https://eng.roborobo.com.kr/main)

activities elevates the potential for robotics activities to generate significant learning opportunities.

It is important for teachers to be aware that experienced programmers likely use previous experiences to outline the general structure of a coding task while novice programmers tend to attack coding tasks without an outline or plan (Robins et al., 2003). Novice programmers will write an opening first line of the code, proceed to write more lines of code until a coding structure appears, and then test the code to see if the structure works. If a programmer is using educational robots, the ability to see immediately if the code works can be both positive and negative. If the code fails, the experienced programmer tends to examine chunks of coding lines to find the cause of the failure; however, the prompt feedback provided by the robot leads novice programmers to engage in unstructured tinkering strategies (e.g., trial and error) as they attempt to fix the code line by line. This trial-and-error approach is not optimal if the goal is to predict outcomes before moving to physical artifacts, just as engineers predict outcomes through mathematical and scientific modeling (Chevalier et al., 2020; Merisio et al., 2021). Although some research studies on the impact of tinkering, a form of trial and error, on student learning report mixed findings (Bevan, 2017; Poce et al., 2019; Vossoughi & Bevan, 2014; Vossoughi et al., 2013), there are studies that have found structured tinkering to have a positive impact on learning, when teacher support keeps the student from becoming frustrated or getting the feeling of being unable to complete the tinkering task (Bevan et al., 2015; Pagano et al., 2020; Simpson et al., 2020). To remain consistent with the principles of engineering design, coding activities should be designed around conceptual learning and not characterized by unsystematic trial and error or memorization of how to build a coding structure. Designing such coding activities is a nontrivial endeavor. Educators in other disciplines (e.g., mathematics education and science education) have developed cognitive demand frameworks to attend to the kinds of thinking students use as they work with disciplinary tasks; this study describes such a framework for examining and developing coding activities associated with robotics.

As early as the 1970's, Halfin (1973) identified the mental processes used by technologists in their work. Wicklein (1993) extended this work for the field of technology education, and Hill (1997) provided a tool and methodology for determining what mental processes were being used by participants in technology education learning activities. Examining these mental processes was not the same as the work mathematics educators were doing with cognitive demand, and yet it was giving attention to what was happening in the minds of learners during instructional activities. Giving attention to what is happening during learning and how learners are engaging with disciplinary concepts and processes is important. Just as metacognition is a process by which we can think about our own thinking, examining mental processes and cognitive demand can allow teachers to think about their teaching in ways that guide its impact and

effectiveness. Teachers who examine what is happening in the minds of learners during classroom activities are guided by a much more effective schema than those who simply follow a curriculum guide or ask students to work a set of problems from a textbook or worksheet.

Instruction in STEM content areas has often relied on memorization of facts and procedures to solve problems or to conduct experiments (Chin et al., 2000; Grove & Bretz, 2012; Hartman & Nelson, 2021). Unless the learning activities provide students with opportunities to engage in reasoning associated with predicting outcomes, the learning activities can be superficial and may not promote long-term retention (Chin et al., 2000; DeDecker et al., 2022). Considering the cognitive demand of classroom tasks redirects attention to the opportunities available for students to learn concepts rather than memorizing procedures.

Related Literature

Tasks and Cognitive Demand

Learning activities, or tasks, structure student thinking around a particular idea within a lesson. Doyle and colleagues began using classroom tasks as a unit of analysis in the 1980s in order to make sense of how students' thinking is structured by classroom events (Doyle, 1988; Doyle & Carter, 1984; Doyle & Sanford, 1985). In the 1990s, mathematics education researchers developed the task analysis guide (Smith & Stein, 1998), using classroom tasks as a unit of analysis, and then science education researchers adapted the task-based framework in the 2010s (Tekkumru-Kisa et al., 2015). Our research team, comprised of experts in mathematics education, science education, engineering, and technology and engineering education, is extending this work to coding tasks. We share the general definition of task presented by Tekkumru-Kisa et al. (2020): "a segment of a classroom activity devoted to the development and assessment of a disciplinary idea and/or practice" (p. 607).

Drawing from the work on cognitive demand in mathematics education (e.g., Stein et al., 1996) and science education (e.g., Tekkumru-Kisa et al., 2015), we define cognitive demand as the intensity of the intellectual work required to complete a disciplinary task. Higher cognitive demand tasks require greater amounts of critical thinking and problem-solving, and lower cognitive demand tasks are often more straightforward and require less mental effort. The level of cognitive demand is an important factor in determining the effectiveness and efficiency of learning, as well as the development of higher-order thinking skills. Cognitive demand is somewhat related to but is distinct from the concept of cognitive load, which has been used within the fields of psychology (see e.g., Plass et al., 2010; Sweller, 1988, 2011) and engineering education (see e.g., Peters, 2015). Cognitive load refers to the amount of mental effort and working memory required by the student to perform the task (Sweller et al., 1998). Cognitive demand shifts the unit of analysis from the individual student to the

task, which can be studied at several phases of a lesson rather than only during task implementation.

This focus on tasks as the unit of analysis is useful because it allows for the study of a wide variety of influences on the cognitive demand of the task throughout all the phases of the lesson. In particular, it can account for the decisions of the teacher before and during the implementation of the task, which may raise, maintain, or lower the cognitive demand. It also provides a mechanism for reflection after implementation.

Cognitive Demand in Mathematics and Science Education

Mathematics education researchers began to examine cognitive demand of mathematics tasks during the 1990s (Stein et al., 1996; Stein & Lane, 1996), and Smith and Stein (1998) developed the Task Analysis Guide (TAG), a framework for analyzing the cognitive demand of mathematics tasks. Their framework described four levels of cognitive demand of tasks: memorization, procedures without connections, procedures with connections, and doing mathematics. The authors identified memorization tasks and procedures without connections tasks as lower cognitive demand. Tasks that fall within these categories often require only the use of memorized facts or procedures without connections to mathematical concepts. Tasks that were categorized as either procedures with connections or doing mathematics were identified as higher cognitive demand. These tasks require attention to mathematical concepts, justifications, multiple representations, and sometimes considerable cognitive effort and self-monitoring.

This framework has been useful for analyzing tasks throughout the instructional process, from the curricular resources that the teachers choose to how the teacher sets up and implements the tasks with students (Arbaugh & Brown, 2005; Boston & Smith, 2009; Henningsen & Stein, 1997; Stein et al., 1996; Stein & Kaufman, 2010). Even when the curricular material provides high-level resources, teachers may lower the cognitive demand during the implementation of the lesson by taking over student thinking (Stein et al., 1996) or by focusing on the accuracy of the procedures rather than on student thinking and reasoning (Henningsen & Stein, 1997). Much work has been done to help teachers select high cognitive demand tasks and support teachers to maintain the cognitive demand during implementation (Arbaugh & Brown, 2005; Boston, 2013; Boston & Smith, 2009; Estrella et al., 2020). Student outcomes based on measures of higher levels of thinking and reasoning have shown greater success when the cognitive demand of the task is maintained throughout the lesson (Boaler & Staples, 2008; Boston & Smith, 2009; Stein & Lane, 1996; Tarr et al., 2008). Several researchers (e.g., Cai et al., 2011; Grouws et al., 2013; Schoenfeld, 2002; Sztajn et al., 2012) have investigated the effects of mathematics curricula intentionally designed with higher-level tasks and found an increase in student achievement, reasoning, and problem-solving.

Tekkumru-Kisa et al. (2015) adapted the mathematics education framework to analyze the cognitive demand of science tasks. Their framework was two-dimensional in that it analyzed (a) the integration of scientific practices and content and (b) the cognitive demand of science tasks. Tekkumru-Kisa et al.'s (2015) framework described the following levels of cognitive demand: memorization, tasks involving scripts, tasks involving guidance for understanding, and doing science. Memorization tasks and tasks involving scripts are considered low cognitive demand. These kinds of tasks focus on the recollection of facts about science content separately from the science practices or on a given step-by-step, recipe-like process. Tasks involving guidance for understanding and doing science tasks are considered high cognitive demand. Tasks involving guidance require students to reason with scientific practices or content with guidance from the task or teacher, whereas doing science tasks require engagement in science practices to make sense of the content. Consistent with work in mathematics education, task analysis during lesson implementation in science classrooms has shown that cognitive demand often decreases during implementation (Kang et al., 2016; Tekkumru-Kisa et al., 2019). Because higher cognitive demand tasks often have some level of ambiguity or uncertainty in the solution or solution process, students unused to these types of problems may become uncomfortable, and teachers may find it difficult to support students, often removing some of the complexity and thereby decreasing the cognitive demand (Henningsen & Stein, 1997; Kang et al., 2016; Tekkumru-Kisa et al., 2019).The work in science education has extended to tracking changes in student thinking throughout the different phases of a task and identifying instructional factors that support maintaining the cognitive demand of tasks (Tekkumru-Kisa et al., 2019; Tekkumru-Kisa et al., 2018).

Research on the cognitive demand of classroom tasks has been beneficial in mathematics and science education. The goal of this paper is to extend the frameworks from these disciplines to offer a new framework for analyzing the cognitive demand of coding tasks.

Method

Researcher Background

This author team consists of individuals from a variety of STEM disciplines who are working together to support and investigate the integration of learning coding and robotics in elementary classrooms. Authors 1, 2, 3, and 4 are mathematics education researchers who worked on the larger project using methods and existing frameworks from mathematics education to investigate classroom discourse and cognitive demand of mathematical tasks. Author 5 is an engineering educator and researcher who seeks to support coding from a young age in hopes of increasing enjoyment of engineering and coding and promote equitable opportunities to pursue STEM disciplines and careers. Author 6 is a technology and workforce education researcher who has deep knowledge of

technology education and research directions and seeks to continue to grow the field using the groundwork laid out by other STEM education disciplines. Although no science educators worked on the writing team, there were science educators within the larger research project who supported and gave direction to the work.

Background

The data for this study came from a larger NSF-funded project called Collective Argumentation and Learning Coding (CALC). The larger study included 32 elementary school teachers enrolled in a graduate-level technology and engineering education course focused on robotics for teachers. In this semester-long professional development course, teachers were introduced to various block-based coding platforms and strategies to integrate coding and robotics across multiple disciplines. Following the professional development course, ten teachers were recruited to participate in the implementation phase of the project, which included classroom observations and collaborative coaching sessions. These ten participating third through fifth grade teachers designed and implemented lessons that incorporated coding and educational robots into regular classroom instruction. The tasks these teachers implemented are the primary source of data from which this framework was developed.

Analysis

From our analysis of these classroom observations, we identified 54 tasks across multiple disciplines including mathematics, science, social studies, and language arts. For this study, we focused on all planned tasks centered on coding content. That is, across all available tasks, we included any task that had a coding focus, and disregarded all tasks that did not include a coding aspect. Forty-one out of the 54 tasks included coding and robotics. We conducted qualitative analysis (Patton, 2015) to develop our framework through iteratively sorting and classifying the tasks according to the cognitive demand of the activities required, using cognitive demand constructs from mathematics and science education to help guide the structure of emerging framework, and then seeking ways of challenging and broadening our developing characteristics and classification by seeking out tasks from outside of the research project.

We began by using an inductive coding process to sort the tasks and identify characteristics of the cognitive activities required to complete the tasks (Schwandt, 2015). The five researchers divided into two groups and sorted the tasks according to how the tasks supported students' thinking about coding, based on intention of the task, as opposed to implementation in the lesson. Each group wrote down characteristics of the tasks based on their groupings and compared ways of distinguishing between tasks. This process allowed for an initial conception of the ways of thinking about coding promoted by these tasks.

After this initial sorting process, we began a deductive coding process by reviewing the mathematics and science cognitive demand frameworks (Smith & Stein, 1998; Tekkumru-Kisa et al., 2015) to structure our thinking about different levels of cognitive demand. We considered the overall structure of these frameworks to consider ways of organizing the characteristics we identified from our initial sorting process. We aimed to create a characterization that differentiated lower cognitive demand coding tasks from those that are higher cognitive demand and to provide descriptive categories within each level. We iteratively sorted the tasks into descriptive categories until our characterizations were robust enough to differentiate between individual tasks. We then examined similarities and differences in the cognitive aspects of the tasks to develop sets of criteria for each category. Our initial categories were low, medium, and high cognitive demand.

To ensure that our criteria for each category were sufficiently broad enough to generalize to coding tasks beyond the tasks generated from our project, we searched for coding tasks on websites[2] containing activities designed for educational robots. We collaboratively examined the tasks from other sources and discussed the level of cognitive demand for these tasks, refining our criteria and expanding to four hypothesized levels.

We used our refined criteria to re-sort the original coding tasks. Three members did this resorting individually, and then checked for consistency with the entire group that resolved any discrepancies together through close examination of the tasks and the criteria. At the end of the resorting process, we examined the tasks and categorizations a final time to check for consistency within and across the four categories. Our refining process led us to generate four categories of cognitive demand for coding tasks: low, medium-low, medium-high, and high (see Table 1). We discuss these categories and their criteria in more detail in the next section.

[2] Each of the websites we surveyed corresponded to a particular educational robot: Ozobot (https://ozobot.com/educate/lessons-and-activities/), Edison (https://meetedison.com/robotics-lesson-plans/), Sphero (https://edu.sphero.com/cwists/category)

Table 1

Task Analysis Guide for Coding (TAG-C)

Level of cognitive demand	Key descriptors
Low	Explicit directions are given to do one (or more) lines of code Task is so open that writing any code would accomplish the task (for example, code your robot to do a dance) Any decisions made by the student are arbitrary (not goal-oriented decisions)
Medium-low	Students have to make some decisions or put some lines of code together to accomplish a task, but the decisions or lines can be accomplished relatively independently, and there is a clear script for accomplishing each part There is little to no consideration of efficiency There is little to no incentive or expectation for creating the entire code before testing each part Some aspects of debugging may be present (for example, in a medium-low coding task, if the teacher asks students to debug their own code, this likely does not raise the cognitive demand of the task)
Medium-high	There is some aspect of metacognition (for instance, by having to think through the situation or problem before writing the code) There are some built-in constraints that require students to engage in some thinking or reflection to put the code together There is consideration of efficiency Some use of control structures (i.e., sequential, selection, repetition) is expected Some expectation for reflecting on their own code Task involves analyzing and comparing or modifying given code

Level of cognitive demand	Key descriptors
High	*Meets all or most of the medium-high descriptors and includes some of the following descriptors:* Creating a program to solve some problem The number of decisions needed is high There are not known clear-cut procedures or scripts to accomplish the task There are potentially sub-routines or coordination of actions The tasks require some complexity in coordinating actions, structures, or routines Students may reflect on previously written code and put it together to create something new

Note. To classify a task, examine its characteristics. Choose the level that includes a majority of its characteristics. A task may fit more than one category; choose the category that includes the preponderance of evidence.

Categories of Cognitive Demand

From our iterative analysis of tasks, we developed a framework for assessing the cognitive demand of coding tasks. In our framework, the Task Analysis Guide for Coding (TAG-C) we identify four descriptive levels (Table 1), similar to Smith and Stein's (1998) Task Analysis Guide (TAG) for mathematics and Tekkumru-Kisa et al.'s (2015) Task Analysis Guide for Science (TAGS). We view these levels as descriptions (Table 2) of the potential cognitive demand of coding and robotics tasks, without intending to place a value judgment on tasks that are "higher" or "lower" cognitive demand. Instead, we intend these levels to be helpful for teachers and curriculum developers in describing the kinds of thinking and reasoning required by tasks, in choosing tasks for specific purposes, and for ensuring that students have access to a range of tasks with different levels of cognitive demand across instructional activities. In the following sections, we describe each level of cognitive demand and exemplify each level with tasks from our project.

Table 2

Classification and Analysis of Example Tasks

Task	Description	Level of cognitive demand
A	Code the Ozobot to perform a dance.	Low
B	Code the Ozobot to travel in a straight line of any distance, turn around, and come back.	Low
C	Draw a polygon and code the Ozobot to travel around the perimeter of your drawn polygon.	Medium-low
D	Code the Edison to travel in a straight line of a particular distance marked by a length of tape on the floor. Reflect on your code: Did it work? If not, how should you change your code?	Medium-low
E	Code the Roborobo to travel around the outline of a capital letter R, drawn on a gridded mat with only line segments and no curves. Write the entire code before implementing it with your robot.	Medium-high
F	Code the Roborobo to travel the path of a square using the loop block. Your code should contain at most six blocks.	Medium-high
G	Measure the side lengths of a given rectangle. Then code the Ozobot to travel around the outline of the rectangle or square on the first try.	Medium-high
H	Code an Ozobot and give it to a partner. The partner must guess the code and recreate the code so their own Ozobot mimics the actions of their partner's robot.	High
I	Code a scene in Scratch that animates all phases of the rain cycle. Explain how you coded your scene.	High

Tasks with Lower Levels of Cognitive Demand

Lower levels of cognitive demand are characterized by fewer opportunities for decision-making, more guidance provided in the task for specific actions, and less consideration of efficiency, control structures, and reflection. There are many reasons one might use a lower cognitive demand task, as described in subsequent sections, but these tasks generally do not provide as much opportunity to develop conceptual understanding of coding or to engage in realistic and problem-centered coding.

Low Cognitive Demand Tasks

A low cognitive demand task has either so many explicit directions that there are no decisions available for students to make or so few directions or expectations that writing almost any code would accomplish the task. These tasks are often used at the very beginning of learning to code to assist students in becoming familiar with the coding interface and language. For instance, when coding Ozobots, some teachers asked their students to code their robots to perform a dance (see Task A in Table 2). This allowed students to use whichever blocks of code they wished to make their robots move, spin, or turn in any direction. While useful, the cognitive demand is classified as low because there is only a loosely defined goal, and it is not necessary for a student to revisit or improve their code, to think about efficiency, or to think through a complex situation prior to writing the code. Similarly, a task that provides explicit directions for coding is also identified as low cognitive demand. For example, in Task B (see Table 2), a teacher asked students to code their Ozobot to travel in a straight line of any distance, turn around, and come back. Each aspect of the task was defined for the students; they did not need to strategize or think through the situation to figure out how to program it. The blocks to be used in Ozoblockly are relatively straightforward to identify; students were learning how to initially put blocks of code together and load it into their Ozobots. Thus, the low cognitive demand of this task allowed for students' attention to be focused on the more logistical tasks of putting the blocks together, loading the program into the robot, and figuring out how to double-click to make the code run.

Medium-low Cognitive Demand Tasks

Medium-low tasks require students to make some decisions in creating their code, and there is a clear script for accomplishing each line of code. When a medium-low task requires multiple lines of code, students can test each line of code independently without having to consider larger chunks of code and why they fit together to accomplish the task. Medium-low tasks do not include considerations of efficiency, creation of larger sections of code without testing, or analysis and modification of code, which are common features that may require students to consider how multiple lines of code fit together. For tasks at

this level, students are expected to write code that correctly accomplishes the given task; there may be some expectation that students debug their own code. In these tasks, because students are not expected to consider larger coding sections at once, this debugging is likely to happen as students try each line of code independently, and thus students may modify their code without considering how the lines of code operate together.

Tasks C and D (see Table 2) are typical examples of medium-low cognitive demand tasks. Task C asks students to first draw a polygon, then code their Ozobots to travel around the outline of the polygon. In this task, students have to make some decisions about their code in terms of programming the Ozobot to go certain lengths and rotate certain angles. However, the lines of code necessary to complete this task can be considered independently, and there are clear scripts for traveling straight lengths and rotating given angles. If students choose a difficult polygon, there may be difficulty in determining lengths and angles, but the code itself is straightforward. For example, there is a block available to code the Ozobot to turn a specific angle, and so they have the line of code as soon as they measure the angle. This task does not require students to write the code for the Ozobot to travel around the outline of the entire polygon before testing it with their robot, and there is not a requirement for them to make their code as efficient as possible. Any debugging that students do will likely be tied to independent lines of code, and so students are unlikely to have to analyze multiple lines of code to make the needed modifications. It is possible that some students may push themselves to write the code more efficiently by using a loop block, for example, if they have several of the same type of movements, or students may try to create the entire code before testing in order to complete the task quickly. However, because there is not an expectation for students to engage in one or more of these ways, the task, as written, is categorized as medium-low.

Task D (see Table 2) requires students to code their Edison to travel a straight distance marked by a length of tape on the floor and to reflect on their code. This task was given to novice coders who were just learning to code the Edison for the first time, and students did not have a memorized routine for approaching this task. When given to novice coders, this task requires that students make some decisions that relate the time delay to the length of tape rather than being directly told what time delay they should write into their code. This feature means that the task contains characteristics of higher cognitive demand tasks. However, there is little opportunity for deep reflection because there is only one line of code to be written, and the decision of the longer or shorter time delay is informed by whether the robot traveled too long or too short of a distance on an initial run. This reflection is not enough for this task to be considered a medium-high cognitive demand task, but the task does an important job of setting up expectations of reflecting and modifying when coding.

Low and medium-low tasks are appropriate as students become familiar with coding robots for the first time or are becoming familiar with a new robot or platform. Lower cognitive demand tasks give students an opportunity to explore and make sense of different features in a platform and experience the outcome of certain codes with the robots. Similarly, medium-low tasks are helpful when students need experience with some basic movements before moving on to more complex tasks or consideration of different control structures. Medium-low cognitive demand coding tasks may also be appropriate when the learning goals of the lesson are not centered around coding. For example, if the learning goal of the lesson is for students to learn the definition of polygon and begin making sense of features of polygons, then it may be helpful for the coding aspect of the task to have medium-low cognitive demand so that the students are able to focus more on the mathematical learning goals.

Tasks with Higher Levels of Cognitive Demand

Medium-high and high cognitive demand tasks can create greater opportunity for students to learn by making connections between prior knowledge and new experiences. These connections can lead to more lasting and meaningful knowledge that students can then draw on in future experiences. These types of tasks can be useful when the goal of a lesson is for students to learn a new coding concept, such as using a repetition control structure to increase efficiency in the code, or for students to begin to make connections across multiple concepts like students would need to do in order to create an animation of the water cycle.

Medium-high Cognitive Demand Tasks

Medium-high cognitive demand coding tasks require students to make decisions and put together multiple lines of code to accomplish a task. Medium-high tasks differ from medium-low tasks in several ways. When a student engages in a medium-high task, there is some expectation of metacognition. For example, the task might require that students think through the situation or problem before writing the code. The requirement to create the entire code before implementing the code with their robot focuses students on the entire process and requires that students anticipate outcomes of their code. For example, students were tasked with programming their robot to travel around the outline of a capital letter R, drawn on a gridded mat and made up of only straight-line segments (see Task E in Table 2). Students were asked to write the entire code before implementing the code with their robot. This requires a level of metacognition and reflection, and it reduces the probability that students will write and test each line of code separately. In medium-high tasks, there is some consideration of efficiency and potentially some use of control structures (i.e., sequential, repetition, selection). Considering how to write more efficient code requires students to consider how different coding structures may reduce the

number of lines of code. This means that students must reflect on aspects of the problem or of their created code that could be modeled or reduced by a control structure. For example, consider a coding task that asks students to program a robot to travel the path of a square. One version of this task might tell students they can use three types of blocks: start, motors, and delay. A subsequent, and more cognitively demanding, version of this task asks students to rewrite their code to program the robot to travel the path of a square using a loop block (see Task F in Table 2). By explicitly asking students to use the loop block, the task meets the criteria for a consideration of efficiency, and in order to consider how to use a loop block, the task requires that students analyze and modify their previously written code. Analyzing and modifying previously written code requires students to engage in some thinking and reflection to put the new code together.

Another example of a medium-high task that requires metacognition and reflection is asking students to program a robot to travel around a square or rectangle with certain parameters and constraints. For example, in a lesson in which students were being introduced to programming a robot, they first programmed the robot to travel 10 steps, then measured (in centimeters) how far the robot traveled. A later task, and one that is considered a task of medium-high cognitive demand, involved using those previous measurements to program the robot to travel around a given rectangle on the first try (see Task G in Table 2). Because students were asked to complete this task on the first try, students engaged in some metacognition and reflection as they used their previous measurements to write code for the robot to trace the square or rectangle. In contrast, a task that appears similar, but is less cognitively demanding, asks students to first draw a polygon and then program the robot to travel the outline of the drawn polygon. This task is considered medium-low and not medium-high for several reasons: a *lack* of parameters or constraints on the drawn polygon, no consideration of efficiency, and no expectation of creating the entire code before testing the program. In this task, students can write and test one line of code at a time. The requirement of "on the first try" in Task G places an emphasis on reflection and accuracy that is not present in medium-low tasks.

High Cognitive Demand Tasks

In Task H (see Table 2), pairs of students were asked to "guess" each other's code. First, one student in the pair coded their Ozobot with any program they preferred, then gave their robot to their partner. The partner was tasked with recreating the code based on the actions of their partner's robot. Coding a robot to mimic a partner's robot's actions requires thought not only about the different coding blocks available but also the order and duration of the blocks to ensure the code produces the same actions of the initial robot. Each pair of students swapped roles so each student had a chance to "guess" their partner's code, and the intention was for the code to be guessed in as few trials as

possible. As students are programming their robot to challenge their partner, they may consider which series of actions might be most difficult to distinguish, and so they may anticipate several outcomes as they are choosing between different options. As they try to solve the challenge presented by their partner, they anticipate which codes produced the outcomes, and then they engage in reflection and modification of their code if their created code does not produce the same robot actions as their partner's code and robot. Students are also trying to get the matching code with as few tries as possible, and likely in fewer tries than their partner takes to guess their written code. Task H requires the student to engage in more metacognition, reflection, and coordination of actions compared to medium-high, medium-low, and low cognitive tasks.

In Task I (see Table 2), students were asked to code a scene in Scratch to animate all phases of the rain cycle (i.e., precipitation, evaporation, condensation, runoff, and collection). Each individual image in Scratch, called a sprite, must be coded individually (e.g., each individual raindrop must be programmed to appear, move, and disappear). In this task, students must code the different phases of the rain cycle sequentially. Coding each of these phases individually is already a complex activity (e.g., using different sprites, costumes, and backgrounds), but having to sequence them to model all phases of the rain cycle requires a high level of complex action coordination. In this scenario the number of choices for students to make is high because of the number of ways that they could choose to portray the rain cycle. The choices are not arbitrary as they will have to correctly model the phenomenon, and one decision about how to portray the cycle leads to subsequent decisions on how to write the code to accomplish the task. These choices often require creative or new coding solutions than what students have previously experienced.

Engaging in high level tasks provides students with opportunities to investigate, strategize, and explore different aspects of coding blocks and their knowledge of coding, which can also be integrated with specific content areas. Teachers can implement varying levels of scaffolding into these lessons to support students without taking away from the overall cognitive demand. For example, in task H (Table 2), the teacher can put a maximum on the number of blocks a student uses to code the original robot. This action is useful for time management, among other things, and does not lower the cognitive demand of the activity. Similarly, the teacher can put parameters around various aspects of Scratch, such as the number of sprites to use, or the different aspects to use, such as sound or text.

Medium-high and high tasks are appropriate when the goal of the lesson is for students to learn or solidify their learning of a specific coding concept. Medium-high cognitive demand tasks give students the chance to reflect and think about their code, and what the student reflects on may be intentionally designed to support their understanding of a specific coding concept. For example, by asking students to create more efficient code using a loop block,

students have the opportunity to learn about and experience the use of control structures. High cognitive demand tasks are appropriate for students to make connections across multiple coding concepts. These tasks can allow students to solidify and extend their knowledge as well as explore new ideas that may be applied or more fully investigated in later lessons. Because high cognitive demand tasks allow for so many student decisions, the goals of the lessons are typically not centered around a single coding concept, but rather the goal of these lessons tend to be more about students engaging in coding practices and connecting multiple ideas.

Interplay of Cognitive Demand, Knowledge, and Grade Level

TAG-C is a general framework that can be used across all grade levels because the cognitive demand of tasks is based on the knowledge and experience level of the students rather than a specific grade level. That is, there can be a medium high task in second grade that looks quite different from a medium-high task in 10th grade. Further, a higher cognitive demand task for a fifth-grade student who is a novice coder may be a lower cognitive demand task for a fourth-grade student who has had more experience coding. For example, consider the medium-high cognitive demand task G where students are asked to make their Ozobot travel the perimeter of a rectangle on the first try. This task requires students to imagine the physical movement around the rectangle, consider the specific measurements, and make connections to specific coding blocks. If students had previous experiences coding the robot to trace shapes, then this task would have been routine, and students would be able to anticipate the connections needed to create the code without engaging in deeper reflection of connections, and thus for these students, the task might be medium-low cognitive demand. Medium-high and high cognitive demand tasks can be used to reinforce knowledge or make more connections; however, if students have enough prior knowledge and experience, then the cognitive demand of the task will be lower than originally intended. This change in cognitive demand can also hold in the other direction; what we may consider lower cognitive demand tasks for experienced coders may be higher cognitive demand for students who have less experience.

Discussion

In ways that are similar to the application of cognitive demand of mathematics and science tasks, TAG-C can allow researchers to analyze the curricular materials used in classrooms and how those tasks support students' higher-level thinking. By focusing on the types of thinking in which students engage within a task, researchers can also begin to study how the cognitive demand of the tasks can change during coding tasks to better understand how teachers can maintain or potentially raise the cognitive demand for students.

Researchers in mathematics and science education have investigated how the cognitive demand of mathematics tasks might be lowered, maintained, or

raised during implementation (Henningsen & Stein, 1997; Stein et al., 1996; Tekkumru-Kisa et al., 2018). In mathematics education, this work supported a better understanding of the ways in which teachers often lower the cognitive demand during a lesson (Henningsen & Stein, 1997), and the work in science education furthered this by making sense of specific ways in which teachers maintain cognitive demand in science lessons (Tekkumru-Kisa et al., 2018). It would be fruitful to explore how the cognitive demand of coding tasks might change during implementation and to examine the role of the teacher in supporting cognitive demand throughout a coding task.

Cognitive demand frameworks have the benefit of being applicable across all levels of schooling and across content areas within a particular subject. For example, the TAG framework (Smith & Stein, 1998) can be applied to counting tasks at the elementary school level as well as geometry tasks at the high school level. Because of this, it is important to consider the prior knowledge and understanding of the students in order to accurately assess the cognitive demand of the task. The counting task that is high cognitive demand for a kindergarten student would likely be a low cognitive demand task for a high school student. Although our framework was developed with coding tasks at the elementary level, we believe it can be applied to coding tasks at higher levels of schooling, including at the undergraduate level. Future research may explore how the TAG-C framework applies at these higher levels, and revisions to the framework may incorporate other constructs, such as levels of abstraction, that were not foregrounded in the set of elementary STEM tasks from which the framework was developed.

While the TAG-C framework is a helpful tool for researchers, it can also support teachers, teacher educators, and curriculum developers. In mathematics education, teachers and teacher educators use the TAG framework (Stein & Smith, 1998) for assessing classroom tasks and discussing how to facilitate higher-level tasks for all students without lowering the cognitive demand. This may mean finding ways to scaffold the task or planning questions to support student thinking during a task. In science education, teacher actions such as advancing students' ideas and pressing for sensemaking support students' higher-level thinking (Tekkumru-Kisa et al., 2019; Tekkumru-Kisa et al., 2018). Teachers can use the TAG-C framework to select and create tasks; teacher educators can use our coding cognitive demand framework to help teachers consider the types of tasks they create, select, and modify for coding lessons and ways of supporting students during those lessons. Professional development centered on supporting teachers' implementation of high cognitive demand tasks have demonstrated an increased focus on teaching actions and on student thinking during lesson implementation (Boston & Smith, 2009; Walkoe, 2015). Researchers in mathematics education have looked at the effects of cognitively demanding tasks within the curricula on student achievement (Cai et al., 2011; Grouws et al., 2013; Schoenfeld, 2002; Sztajn et al., 2012), and the TAG-C

framework can be used in a similar way to develop and assess curriculum for learning coding. Curriculum developers can use this framework to consider the cognitive demand of their tasks and how they support students' higher-level thinking, and then they can research the effects of such a curriculum on student achievement.

Differences in student ability and experiences are common in coding contexts and create an important challenge for both teachers and curriculum developers. Even by 5th grade, there are often students who have studied coding independently and through schooling since first or second grade, and within those same classrooms are completely novice coders. A task that is high cognitive demand for the novice coders may be lower cognitive demand for experienced coders in the same classroom. Differences in student experiences and prior knowledge are not uncommon for any discipline, and so differentiation is just as important in coding lessons as it is in science, reading, or math. Higher cognitive demand tasks frequently offer more opportunities for students to make decisions and for instructors to change parameters to fit the needs of students, and so these tasks can offer greater opportunities for differentiation within a specific lesson. Strong educational curriculum should provide ways of differentiating activities for students without lowering the cognitive demand of the activity. Lower cognitive demand tasks are typically more explicit and often do not have as many opportunities for students to make decisions, and so it may be more difficult to provide differentiation within a single activity. In these cases, it may be worthwhile to consider alternative activities with the appropriate challenge for students who already have the requisite knowledge.

Considering curriculum development leads to the practical and open question of how much of each of these different levels is optimal for student learning. We raise some considerations with our framework about when different levels of tasks may be useful. For example, lower cognitive demand tasks may be useful when students are learning a new coding platform and need to spend cognitive effort on non-coding aspects of the task, such as learning about the physical characteristics of their robot or the technicalities of loading a given code into the robot. However, within the same lesson, students could complete a high-cognitive demand task such as the "Guess my code" task, and so it may be beneficial to research how much instructional time spent on lower versus higher cognitive demand tasks is beneficial for students, and what types of learning outcomes may be supported by different distributions of tasks. This question has not been thoroughly investigated in either math or science education, and there is a need for more research in this area across disciplines.

Finally, we suggest that this type of framework that assesses cognitive demand could be usefully developed and applied to other disciplines. For example, a cognitive demand framework could be developed for engineering tasks, and research with this framework could provide useful information for engineering task development and implementation in the same ways that it has

served mathematics and science education. A cognitive demand framework is widely useful because of its general application to tasks without being specific to grade-level or domain within a given discipline, and the development of the framework is even more worthwhile because of its usefulness for teachers and researchers as well as other educational stakeholders.

In our project, the teachers completed some lessons that were focused on coding, but more often they worked on integrating robotics and coding with other disciplines. The integration of coding tasks with other disciplines increases the complexity of assessing the cognitive demand of tasks, and this also increases the difficulty of understanding how to best support students during implementation of the tasks. More work needs to be done to consider the cognitive demand of integrated tasks that include coding.

References

Arbaugh, F., & Brown, C. A. (2005). Analyzing mathematical tasks: A catalyst for change? *Journal of Mathematics Teacher Education*, *8*, 499-536. https://doi.org/10.1007/s10857-006-6585-3

Bevan, B. (2017). The promise and the promises of making in science education. *Studies in Science Education*, *53*(1), 75-103. https://doi.org/10.1080/03057267.2016.1275380

Bevan, B., Gutwill, J. P., Petrich, M., & Wilkinson, K. (2015). Learning through STEM-rich tinkering: Findings from a jointly negotiated research project taken up in practice. *Science Education*, *99*(1), 98-120. https://doi.org/10.1002/sce.21151

Boaler, J., & Staples, M. (2008). Creating mathematical futures through an equitable teaching approach: The case of Railside School. *Teachers College Record*, *110*(3), 608-645. https://doi.org/10.1177/016146810811000302

Boston, M. D. (2013). Connecting changes in secondary mathematics teachers' knowledge to their experiences in a professional development workshop. *Journal of Mathematics Teacher Education*, *16*, 7-31. https://doi.org/10.1007/s10857-012-9211-6

Boston, M. D., & Smith, M. S. (2009). Transforming secondary mathematics teaching: Increasing the cognitive demands of instructional tasks used in teachers' classrooms. *Journal for Research in Mathematics Education*, *40*(2), 119-156. https://doi.org/10.2307/40539329

Bransford, J., Vye, N., & Bateman, H. (2002). Creating high-quality learning environments: Guidelines from research on how people learn. In P. A. Graham & N. Stacey (Eds.), *The knowledge economy and postsecondary education: Report of a workshop*. (pp. 159-198) Washington, D.C.: National Academy Press. https://doi.org/10.17226/10239

Bureau of Labor Statistics. (2022, September 8). *Employment by detailed occupation*. https://www.bls.gov/emp/tables/emp-by-detailed-occupation.htm

Cai, J., Wang, N., Moyer, J. C., Wang, C., & Nie, B. (2011). Longitudinal investigation of the curricular effect: An analysis of student learning outcomes from the LieCal Project in the United States. *International Journal of Educational Research*, *50*(2), 117-136. https://doi.org/10.1016/j.ijer.2011.06.006

Chevalier, M., Giang, C., Piatti, A., & Mondada, F. (2020). Fostering computational thinking through educational robotics: A model for creative computational problem solving. *International Journal of STEM Education*, *7*(1), 1-18. https://doi.org/10.1186/s40594-020-00238-z

Chin, C., & Brown, D. E. (2000). Learning in science: A comparison of deep and surface approaches. *Journal of Research in Science Teaching, 37*(2), 109-138. https://doi.org/10.1002/(SICI)1098-2736(200002)37:2%3C109::AID-TEA3%3E3.0.CO;2-7

DeDecker, S., Chouvalova, A., Gordon, K., Clemmer, R., & Vale, J. (2022). Memorization: Friend or foe when solving problems in STEM undergraduate courses. *Proceedings of the Canadian Engineering Education Association (CEEA).* https://doi.org/10.24908/pceea.vi.15945

Doyle, W. (1988). Work in mathematics classes: The context of students' thinking during instruction. *Educational Psychologist, 23*, 167–180. https://doi.org/10.1207/s15326985ep2302_6

Doyle, W., & Carter, K. (1984). Academic tasks in classrooms. *Curriculum Inquiry*, *14*(2), 129–149. https://doi.org/10.2307/3202177

Doyle, W., & Sanford, J. P. (1985). *Managing students' work in secondary classrooms: Practical lessons from a study of classroom tasks* (Report No. RDCTE-6193). Austin: Research and Development Center for Teacher Education, University of Texas at Austin. https://eric.ed.gov/?id=ED271319

Early, D. M., Rogge, R. D., & Deci, E. L. (2014). Engagement, alignment, and rigor as vital signs of high-quality instruction: A classroom visit protocol for instructional improvement and research. *The High School Journal*, *97*(4), 219–239. http://www.jstor.org/stable/43281032

Estrella, S., Zakaryan, D., Olfos, R., & Espinoza, G. (2020). How teachers learn to maintain the cognitive demand of tasks through Lesson Study. *Journal of Mathematics Teacher Education, 23*, 293-310. https://doi.org/10.1007/s10857-018-09423-y

Grove, N. P., & Bretz, S. L. (2012). A continuum of learning: from rote memorization to meaningful learning in organic chemistry. *Chemistry Education Research and Practice, 13*(3), 201-208. https://doi.org/10.1039/C1RP90069B

Grouws, D. A., Tarr, J. E., Chávez, Ó., Sears, R., Soria, V. M., & Taylan, R. D. (2013). Curriculum and implementation effects on high school students' mathematics learning from curricula representing subject-specific and integrated content organizations. *Journal for Research in Mathematics Education, 44*(2), 416-463. https://doi.org/10.5951/jresematheduc.44.2.0416

Halfin, H. H. (1973). *Technology: A process approach* (Unpublished doctoral dissertation). West Virginia University, Morgantown.

Hartman, J. R., & Nelson, E. A. (2021). *A paradigm shift: The implications of working memory limits for physics and chemistry instruction.* https://doi.org/10.48550/arXiv.2102.00454

Henningsen, M., & Stein, M. K. (1997). Mathematical tasks and student cognition: Classroom-based factors that support and inhibit high-level mathematical thinking and reasoning. *Journal for Research in Mathematics Education, 28*(5), 524-549. https://doi.org/10.5951/jresematheduc.28.5.0524

Hill, R. (1997). The design of an instrument to assess problem solving activities in technology education. *Journal of Technology Education, 9(*1), 31-46.

Hill, R. B. (2006). New perspectives: Technology teacher education and engineering design. *Journal of Industrial Teacher Education, 43*(3), 45.

International Technology and Engineering Educators Association. (n.d.). *Technology and Engineering Education Collegiate Association (TEECA)*. Retrieved from https://www.iteea.org/About/Leadership/40079/TEECA.aspx#tabs

International Technology and Engineering Educators Association. (2020). *Standards for technological and engineering literacy: The role of technology and engineering in STEM education.* www.iteea.org/STEL.aspx

International Technology and Engineering Educators Association. (2021). *Engineering byDesign*. https://www.iteea.org/File.aspx?id=111278&v=5c360a06

Kang, H., Windschitl, M., Stroupe, D., & Thompson, J. (2016). Designing, launching, and implementing high quality learning opportunities for students that advance scientific thinking. *Journal of Research in Science Teaching, 53*(9), 1316-1340. https://doi.org/10.1002/tea.21329

Merisio, C., Bozzi, G., Datteri, E. (2021). There is no such thing as a "Trial and error strategy". In M. Malvezzi, D. Alimisis, M. Moro, (Eds.), *Education in & with Robotics to Foster 21st-Century Skills: Proceedings of EDUROBOTICS 2020* (pp. 190-201). Springer, Cham. https://doi.org/10.1007/978-3-030-77022-8_17

Pagano, L. C., Haden, C. A., & Uttal, D. H. (2020). Museum program design supports parent–child engineering talk during tinkering and reminiscing. *Journal of Experimental Child Psychology, 200*, https://doi.org/10.1016/j.jecp.2020.104944

Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1986). Conditions of learning in novice programmers. *Journal of Educational Computing Research, 2*(1), 37-55. https://doi.org/10.2190/GUJT-JCBJ-Q6QU-Q9PL

Peters, M. (2015). Using cognitive load theory to interpret student difficulties with a problem-based learning approach to engineering education: a case

study. *Teaching Mathematics and its Applications: An International Journal of the IMA*, *34*(1), 53-62. https://doi.org/10.1093/teamat/hru031

Patton, M. Q. (2015). *Qualitative research and evaluation methods* (4th ed.). Sage.

Plass, J. L., Moreno, R., & Brünken, R. (Eds.). (2010). *Cognitive load theory.* Cambridge University Press. https://doi.org/10.1017/CBO9780511844744

Poce, A., Amenduni, F., & De Medio, C. (2019). From tinkering to thinkering. Tinkering as critical and creative thinking enhancer. *Journal of e-Learning and Knowledge Society*, *15*(2). https://doi.org/10.20368/1971-8829/1639

Rojewski, J. W., & Hill, R. B. (2014). Positioning research and practice in career and technical education: A framework for college and career preparation in the 21st century. *Career and Technical Education Research, 39*(2), 137-150. https://doi.org/10.5328/cter39.2.137

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137-172. https://doi.org/10.1076/csed.13.2.137.14200

Schoenfeld, A. H. (2002). Making mathematics work for all children: Issues of standards, testing, and equity. *Educational Researcher, 31,* 13–25. https://doi.org/10.3102/0013189X031001013

Schwandt, T. A. (2015). *The SAGE dictionary of qualitative inquiry* (4th ed.). Sage.

Simpson, A., Burris, A., & Maltese, A. (2020). Youth's engagement as scientists and engineers in an afterschool making and tinkering program. *Research in Science Education, 50*(1), 1-22. https://doi.org/10.1007/s11165-017-9678-3

Smith, M. S., & Stein, M. K. (1998). Reflections on practice: Selecting and creating mathematical tasks: From research to practice. *Mathematics Teaching in the Middle School, 3*(5), 344-350. https://doi.org/10.5951/MTMS.3.5.0344

Stein, M. K., Grover, B. W., & Henningsen, M. (1996). Building student capacity for mathematical thinking and reasoning: An analysis of mathematical tasks used in reform classrooms. *American Educational Research Journal*, *33*(2), 455-488. https://doi.org/10.3102/00028312033002455

Stein, M. K., & Kaufman, J. H. (2010). Selecting and supporting the use of mathematics curricula at scale. *American Educational Research Journal*, *47*(3), 663-693. https://doi.org/10.3102/0002831209361210

Stein, M. K., & Lane, S. (1996). Instructional tasks and the development of student capacity to think and reason: An analysis of the relationship between teaching and learning in a reform mathematics project. *Educational Research and Evaluation*, *2*(1), 50-80. https://doi.org/10.1080/1380361960020103

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, *12*(2), 257-285. https://doi.org/10.1016/0364-0213(88)90023-7

Sweller, J. (2011). Cognitive load theory. In *Psychology of learning and motivation* (Vol. 55, pp. 37-76). Academic Press. https://doi.org/10.1016/B978-0-12-387691-1.00002-8

Sweller, J., van Merriënboer, J. J. G., & Paas, F. (1998). Cognitive architecture and instructional design. *Educational Psychology Review, 10*, 251–296. https://www.jstor.org/stable/23359412

Sztajn, P., Confrey, J., Wilson, P. H., & Edgington, C. (2012). Learning trajectory based instruction: Toward a theory of teaching. *Educational Researcher*, *41*(5), 147–156. https://doi.org/10.3102/0013189X12442801

Tarr, J. E., Reys, R. E., Reys, B. J., Chávez, Ó., Shih, J., & Osterlind, S. J. (2008). The impact of middle-grades mathematics curricula and the classroom learning environment on student achievement. *Journal for Research in Mathematics Education*, *39*(3), 247-280. https://doi.org/10.2307/30034970

Tekkumru-Kisa, M., Schunn, C., Stein, M. K., & Reynolds, B. (2019). Change in thinking demands for students across the phases of a science task: An exploratory study. *Research in Science Education*, *49*, 859-883. https://doi.org/10.1007/s11165-017-9645-z

Tekkumru-Kisa, M., Stein, M. K., & Coker, R. (2018). Teachers' learning to facilitate high-level student thinking: Impact of a video-based professional development. *Journal of Research in Science Teaching*, *55*(4), 479-502. https://doi.org/10.1002/tea.21427

Tekkumru-Kisa, M., Stein, M. K., & Doyle, W. (2020). Theory and research on tasks revisited: Task as a context for students' thinking in the era of ambitious reforms in mathematics and science. *Educational Researcher, 49*(8), 606-617. https://doi.org/10.3102/0013189X20932480

Tekkumru-Kisa, M., Stein, M. K., & Schunn, C. (2015). A framework for analyzing cognitive demand and content-practices integration: Task analysis guide in science. *Journal of Research in Science Teaching*, *52*(5), 659-685. https://doi.org/10.1002/tea.21208

Vossoughi, S., & Bevan, B. (2014). Making and tinkering: A review of the literature. *National Research Council Committee on Out of School Time STEM, 67*, 1-55.

Vossoughi, S., Escudé, M., Kong, F., & Hooper, P. (2013). Tinkering, learning & equity in the after-school setting. In A*nnual FabLearn conference. Palo Alto, CA: Stanford University*.

Walkoe, J. (2015). Exploring teacher noticing of student algebraic thinking in a video club. *Journal of Mathematics Teacher Education*, *18*, 523-550. https://doi.org/10.1007/s10857-014-9289-0

Weiss, I. R., & Pasley, J. D. (2004). What is high-quality instruction? *Educational Leadership*, *61*(5), 24–28. https://eric.ed.gov/?id=EJ716718

Wicklein, R. C. (1993). Developing goals and objectives for a process-based technology education curriculum. *Journal of Industrial Teacher Education 30*(3), 66-80. https://eric.ed.gov/?id=EJ463556

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends, 60*, 565-568. https://doi.org/10.1007/s11528-016-0087-7

About the Authors

Anna Bloodworth (anna.bloodworth@uga.edu) is a Graduate Research Assistant in the Department of Mathematics, Science, and Social Studies Education at the University of Georgia. https://orcid.org/0009-0006-7566-0638

AnnaMarie Conner (aconner@uga.edu) is a Professor of Mathematics Education in the Mary Frances Early College of Education at the University of Georgia. https://orcid.org/0000-0001-5510-0795

Claire Miller (clairemiller@uga.edu) is a Graduate Research Assistant in the Department of Mathematics, Science, and Social Studies Education at the University of Georgia. https://orcid.org/0000-0002-5559-4921

Lorraine Franco (lorraine.franco@uga.edu) is a Graduate Research Assistant in the Department of Mathematics, Science, and Social Studies Education at the University of Georgia. https://orcid.org/0000-0001-6663-5795

Timothy Foutz (tfoutz@uga.edu) is a Professor of Engineering in the College of Engineering at the University of Georgia. https://orcid.org/0000-0001-9102-6203

Roger Hill (rbhill@uga.edu) is a Professor of Workforce Education in the Mary Frances Early College of Education at the University of Georgia. https://orcid.org/0000-0002-9402-5676