

Journal of Educational Technology Development and Exchange (JETDE)

Volume 14 | Issue 1

2021

Examining Trajectories of Elementary Students' Computational Thinking Development Through Collaborative Problem-Solving Process in a STEM-Integrated Robotics Program

Yi-Chun Hong
Arizona State University, yhong22@asu.edu

Yingxiao Qian
Arizona State University, yqian36@asu.edu

Yu-Fen Yang
National Yunlin University of Science and Technology, yangy@yuntech.edu.tw

Follow this and additional works at: <https://aquila.usm.edu/jetde>



Part of the [Educational Technology Commons](#), and the [Elementary Education Commons](#)

Recommended Citation

Hong, Yi-Chun; Qian, Yingxiao; and Yang, Yu-Fen (2021) "Examining Trajectories of Elementary Students' Computational Thinking Development Through Collaborative Problem-Solving Process in a STEM-Integrated Robotics Program," *Journal of Educational Technology Development and Exchange (JETDE)*: Vol. 14 : Iss. 1 , Article 2.

Available at: <https://aquila.usm.edu/jetde/vol14/iss1/2>

This Article is brought to you for free and open access by The Aquila Digital Community. It has been accepted for inclusion in Journal of Educational Technology Development and Exchange (JETDE) by an authorized editor of The Aquila Digital Community. For more information, please contact Joshua.Cromwell@usm.edu.

Examining Trajectories of Elementary Students' Computational Thinking Development Through Collaborative Problem-Solving Process in a STEM-Integrated Robotics Program

Cover Page Footnote

Authors declare that they have no conflicts of interest. This study has been approved by the IRB of the University of Georgia (IRB Approval #: 000001432) and informed consent was obtained from all participants in the study. The datasets used in this study are unavailable due to privacy and data protection concerns.

Examining Trajectories of Elementary Students' Computational Thinking Development Through Collaborative Problem-Solving Process in a STEM-Integrated Robotics Program

Yi-Chun Hong

Arizona State University

Yingxiao Qian

Arizona State University

Yu-Fen Yang

National Yunlin University of Science and Technology

Abstract: *Developing K-12 students' computational thinking (CT) skills is essential. Building on the existing literature that has emphasized programming skill development, this study expands the focus to examine students' use of underlying CT cognitive skills during collaborative problem-solving processes. A case study approach was employed to examine video data of 5th graders engaging in an integrated-STEM robotics curriculum. The findings reveal that students applied algorithmic thinking most frequently and prediction the least. They recorded most debugging behaviors initially in the problem-solving process, but after accumulating more experiences their uses of other CT skills, including algorithmic thinking, pattern recognition, and prediction, increased. Implications for developing young learners' CT skills to solve real-world problems are discussed.*

Keywords: computational thinking, STEM education, robotics, collaborative problem solving, cognitive skills

Introduction

The STEM (science, technology, engineering, and mathematics) workforce is central to a country's national economy and global competitiveness. Students with computational thinking (CT) ability are more likely to become STEM literate, prepared for a future career in STEM fields (Weintrop et al., 2016). The need to train students with computational thinking (CT) is on the rise. CT is a thinking process that enables an individual to identify and solve a real-world problem by leveraging the synergy between humans and machines in order to automate the process to achieve an efficient and effective solution (Wing, 2011). Many educators interpret CT as simply programming skills and several CT studies have aimed at improving learners' programming skills using different instructional interventions (e.g., Dohn, 2020). However, CT includes a broad range of cognitive skills beyond programming (Shute et al., 2017; Wing, 2011). For example, Shute and colleagues (2017) reviewed and identified the components of CT that were shared across different models: "decomposition, abstraction, algorithms, and debugging" (p. 145). Nonetheless, current understanding of how individuals employ these cognitive skills when applying CT to solve problems remains mostly conceptual or stays within the context of programming education. Empirical evidence on whether and how these CT cognitive skills are exercised during the problem-solving process will allow us to better support students' CT development.

Collaboration and communication are essential skills for applying CT to solve a problem (ISTE & CSTA, 2011). CT often involves individuals collaborating in team problem-solving processes as they

solve problems, build algorithms or rules, troubleshoot, and create models (Berland & Lee, 2011; NRC, 2010). To teach CT, many teachers implement teamwork as a common instructional activity designed to support students' development of CT skills (Bower et al., 2017). However, the existing literature mostly reveals its frequent adoption but has not yet provided an in-depth and qualitative understanding of how students exercises CT sub-skills over the course of collaborative learning processes. Therefore, this study was conducted to uncover different CT sub-skills present in student dialogues when they engaged in collaborative problem-solving processes in a STEM-integrated robotics program.

Literature Review

Computational Thinking Sub-Skills

The burgeoning wave of computing in modern society has manifested the need to prepare individuals with CT skills, which is comparable to the other essential literacy skills of reading, writing, and arithmetic. Moving beyond the interpretation of CT as simply programming skills, Wing (2011) first defined CT as a generic problem-solving process and later further clarified that "computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (p.1). Echoing this conception, Tinker construed that CT allows a problem solver to decompose a problem and to identify solutions that can function automatically and efficiently to the subsets of the problem (NRC, 2010). CT is perceived as a collection of cognitive skills that enables individuals to solve problems

effectively. Sengupta and colleagues (2013) have described CT practice as including “problem representation, abstraction, decomposition, simulation, verification, and prediction” (p. 351). Grover and Pea (2013) also identified several elements of CT: “abstraction and pattern generalizations; systematic processing of information; symbol systems and representations; algorithmic notions of flow of control, structured problem decomposition; iterative, recursive, and parallel thinking; conditional logic; efficiency and performance constraints; debugging and error systematic detection” (pp.39-40). In a literature review of CT, Shute and colleagues (2017) summarized the following as commonly described CT components: “decomposition, abstraction, algorithms, and debugging” (p. 145). These various conceptual discussions from the existing literature, despite differences in their identifications of CT elements, illustrate the need for multiple cognitive skills to operate together in order to enact CT for solving problems. Even with distinct identifications of CT elements by different scholars, we observe the following overlapping cognitive skills and use them to examine students’ CT development in our investigation: (1) debugging, (2) algorithmic thinking, (3) decomposition, (4) abstract thinking, (5) pattern recognition, and (6) prediction (Grover & Pea, 2013; NRC, 2010; Sengupta et al., 2013; Shute et al., 2017; Wing, 2011).

Several cognitive capacities are crucial for applying CT in a problem-solving process. However, the current understanding of relevant CT sub-skills comes mostly from the study of improvement in programming skills (Dohn, 2020). A review of programming education by Lye and Koh (2014) revealed that most studies investigated student acquisition of

programming concepts and only a few looked into programming practices, with most of those focusing on debugging behaviors, which is the first CT sub-skill investigated in this study.

The second critical CT cognitive element, algorithmic thinking, has also been studied mainly in the context of programming education (Grover et al., 2015). Algorithmic thinking allows problem solvers to plan and identify a series of steps to arrive at a solution. It is challenging for students to acquire algorithmic thinking skills, so research has focused on implementing learning activities that support the development of such skills (e.g., Grover et al., 2015).

The third cognitive capacity needed for processing CT is decomposition. There are numerous studies examining students’ ability to decompose programming tasks (e.g., Chao, 2016; Kwon & Cheon, 2019). Kwon and Cheon (2019) revealed that most students experienced difficulty decomposing more complex and larger problems. Additionally, Chao (2016) found that students adopting the selective approach (i.e., using if-then functions for a programming task) decomposed the problem into coarse-grained subparts, and then generated a more efficient solution. Conversely, students tackling the programming task in a linear manner, decomposed problems into a fine-grained sub-parts, and then produced relatively inefficient solutions.

The fourth element, abstraction, is a cognitive skill that allows one to remove irrelevant information and variables, and concentrate only on the essence of the problem (Grover & Pea, 2013). Zhao and Shute (2019) studied eighth-grade students’ abstractive thinking in a block-based programming game,

Penguin Go. During the game, students were expected to disregard extraneous disparities to grasp the essential commonalities shared among various objects and/or procedures. Nevertheless, they demonstrated limited abstractive thinking because most of them applied a trial-and-error approach rather than identified the essence of the problem.

Another cognitive skill essential to effective CT is pattern recognition, which enables problem solvers to retrieve and reuse a solution (i.e., patterns) from an analogous problem (Grover & Pea, 2013). Similar to the other cognitive skills, a few studies have been conducted in the programming context to examine how students repeatedly use generalizable code blocks in an algorithmic solution. Kwon and Cheon (2019) investigated middle school students' capacities of pattern recognition by evaluating whether they could identify generalizing solutions by "parameterizing the variables" (e.g., using a repeated block in Scratch programming projects). The results showed that among seven students only one exerted pattern recognition and demonstrated more advanced CT, especially in using logic and loops to solve multiple analogous problems. However, there was no explicit explanation as to how students' pattern recognition was used in programming or other computing tasks.

The last CT element, prediction, is a crucial skill for professionals in computer science and STEM domains (Bers, 2017). In computer science, programmers need to be able to predict an outcome from several lines of code. In STEM fields, the ability to predict enables individuals to draw inferences and anticipate what will happen based on the collected data. For example, scientists can predict when storms will hit a city based

on weather data. Existing literature has also considered prediction as a major component of CT. However, there is a lack of empirical study with a focus on investigating K-12 students' prediction in CT contexts (Sengupta et al., 2013).

According to our review of the existing literature, the research on prediction remains at the conceptual level, while research on the other skills is in connection with computer programming education. As many scholars have argued, CT education should be broadened to develop students' ability to apply CT for solving problems in other contexts, such as STEM fields, rather than being limited to training programmers (Caeli & Bundsgaard, 2020; Roman-Gonzalez et al., 2017). Furthermore, the majority of the existing empirical studies inspected only one CT sub-skill (e.g., Chao, 2016; Zhao & Shute, 2019). Despite those results providing an understanding on how students exercised a specific cognitive skill, further studies are called for to explore the interactions among CT skills. CT entails the interaction of multiple elements (e.g., Grover & Pea, 2013; Shute et al., 2017) and the effects of individual skills may interact with each other and affects overall CT ability. Thus, our study aims to explore CT sub-skills are present in students' problem-solving process and the possible interactions among the different CT sub-skills.

Collaboration of Computational Thinking

Social interactions are central to the development of CT ability (NRC, 2010). Collaboration allows students to engage in active and constructive learning while problem-solving. Working in a team allows individuals to "develop representations, debug processes, and so on, resulting in a

collaborative process of discovery that is richer than that of any single individual” (NRC, 2010, p. 27). Group work is one of the common instructional activities that teachers use in CT education, particularly programming education (Bower et al., 2017). Pairs are able to detect any errors early, strengthen their understanding of knowledge, and present multiple perspectives to solve problems (Iiskala et al., 2011; Manlove et al., 2006). Recognizing the benefits of collaborative problem-solving, numerous studies compared programming done alone with programming done with partners at both college and K-12 levels. They found that students learned better, and with more enjoyment, when engaging in pair programming (Zhong, et al., 2016). One study on student discourse in a game-based, CT-focused environment revealed that conversations among team members occurred frequently, as they figure out strategies to advance their moves in the game (Berland & Lee, 2011). Student collaborative problem-solving was also effective for developing graphics through coding (Doleck et al., 2017). These studies show the effectiveness of collaborative problem solving.

Building upon these promises, this study seeks to further our knowledge about CT by exploring how students exercise CT-related cognitive skills during collaborative problem-solving processes. To this end, two research questions guide this investigation:

1. What is the pattern of students’ CT sub-skills that are involved in the collaborative problem-solving process?
2. What are the trajectories of CT skills

exercised over the course of collaborative problem-solving process between pairs?

Methodology

Research Design & Participants

For this case study, we reached out to the STEM Integration Coordinator at a local school district in the southeastern United States to recruit 5th grade teachers to implement a STEM-integrated robotics curriculum, *Danger Zones*, in the classroom setting. Students were 10-11 years old, on average. They were placed into pairs to collaborate for problem-solving tasks. With the nature of case study, three pairs of volunteering students were selected to be the focus of our investigation, with two pairs consisting of a boy and a girl and a pair of boys. Each pair’s collaborative problem-solving behaviors were video recorded.

Curriculum Design


The STEM-integrated robotics curriculum, *Danger Zones*, required students to collaboratively solve two tasks using knowledge from STEM domains, programming skills, and different CT cognitive skills. Students were introduced to a scenario where they were charged to design a robot to assist scientists in collecting three research samples in an active volcanic area (See Figure 1). The curriculum lasted two weeks and consisted of ten 50-minute sessions. A detailed description of the curriculum is presented in Figure 1.

Figure 1. Learning activities in a STEM-integrated robotics curriculum

Lesson 1 Danger Zone

- Learn the science content related to volcanoes
- Learn engineering design process
- Learn the principles for building a collaborative relationship with peers

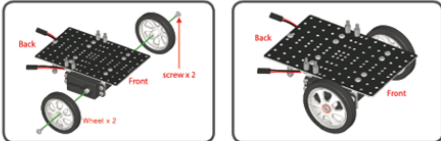
Scenario



The United States Geological Survey, or USGS, wants to study a volcano that has recently begun to show signs of activity. As part of their study, they need to collect samples from the area. However, the volcano is emitting gases that make the area too dangerous for humans to enter.

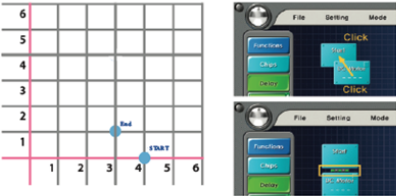
Lesson 2 Build-a-Bot

- Learn the key components of robots
- Assemble robot



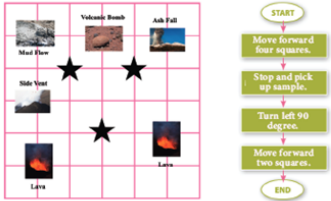
Lesson 3 Primary Programming

- Learn the interface of graphical programming software and programming logic
- Task 1: Design the robot to move forward for one square and turn at a 90-degree angle



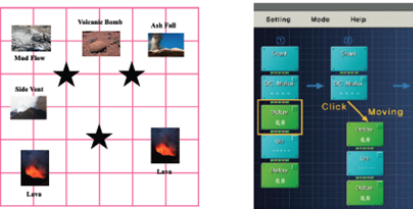
Lesson 4 Purposeful Programming

- Review mathematical concepts: decimals, division
- Task 2: Engage in the design process to design a robot path and program the robot to stop and collect three samples



Lesson 5 Prime Optimization


- Review mathematical concepts: measurement, coordinates
- Task 2: Engage in the design process to design a robot path and program the robot to stop and collect three samples



Lesson 6 Share

- Present final outcomes and reflect

Essential Question



How does the engineering design process help with problem solving?

Data Collection and Coding Scheme

The data were the audio files from the four-hour long conversations recorded for each of the three pairs during Lessons Three

to Five. Conversations were first transcribed verbatim before being coded according to the six CT sub-skills that we found to be common in the existing literature. These skills are defined, with an example for each, in Table 1.

Table 1. Coding scheme of CT sub-skills

Code	Definition	Example Quotes
Abstraction	Involves hiding unnecessary or irrelevant details to focus only on the most relevant information to simplify complexity.	N/A
Algorithmic thinking	Involves the construction of a set of steps to solve a specific problem. Algorithms are used to calculate speed rate, to plan the route, or to write a program that could tell the robots to move exactly as we want it to.	“[The robot] turned not enough. How about we add something on [the delay chip].” “We need to set up .25 because we’re going to from here to be there.”
Debugging	A process of figuring out unexpected outcomes. In computer science, debugging is viewed as a specific process of detecting errors while programming.	“Oh we forgot to put time there!” “[The robot moved] a little too far.”
Pattern recognition	A process of identifying the common features of a problem, and then retrieving and reusing prior solutions to resolve analogous problems.	“Put the same time [in programming]. Just use the same time [setting].” “You should know, we used 3.4 [to have the robot move certain distance], this one is still [3.4].”
Problem decomposition	A process of breaking a complicated problem into smaller and more tractable pieces that are easier to deal with (e.g., modularizing).	“Now we need to turn. Then go left. Then we need to go forward.”
Prediction	The ability of envisioning a particular action of the robot based on what students have programmed.	“Then it [the robot] would just go forward, go backwards, go forward again, turn right and then go forward.” “We stopped right. Then we stopped right there, then we turned to the left, and now we need to go forward for one second.”

Data Analysis

After examining the transcripts, the data were analyzed using content analysis to investigate the frequencies of the different six skills of CT occurring within the conversations. The coding process involved two researchers. They first reviewed the transcripts with the coding scheme in mind. Prior to coding, they discussed and reached consensus on the unit of analysis by segmenting the conversations produced by each pair of students to each testing event (Chi, 1997). A testing event is defined as the time when the students began to interpret the problem, generated a solution, programmed using the provided software, to the time when they were ready for testing the programming at the testing station. Using this approach, the data could be coded, within the context, as an event (Chi, 1997). Each event could entail multiple codes that represent several sub-skills of CT produced by each participant.

The coding included two rounds, individual and then collective. During the first round, each researcher individually read through the transcripts to become familiar with them. Then each researcher started coding independently the transcripts from the first pair of students' conversations, event by event, in reference to the six CT sub-skills (in Table 1). The goal of this step was to assign relevant codes in each event. Each researcher reviewed the coded transcripts iteratively until satisfied. The two coders then reviewed both of their codes and discussed until they reached 100% agreement. They then used their emerging understanding to independently code the conversations generated by the other two pairs. The inter-rater reliability between the two raters was 88%.

Findings

Overall Pattern of CT Sub-Skills Involved in the Collaborative Problem-Solving Process

The most frequently observed skill was algorithmic thinking (131), which occurred in almost all events (Table 2). Abstraction was the next most frequently (89) noted cognitive skill. Given that participants were explicitly told with all essential information of the tasks they solved, their dialogues showed that their focus on relevant information was on target in each event. In contrast, prediction was the least frequently (28) observed. We examined the CT components evidenced in respective lessons. In Lesson 3, there were more conversations related to algorithmic thinking (35) and debugging (29). This lesson offered students their first opportunity to learn to use the software and programming logic, and thus they mainly adopted a trial-and-error approach to perform Task 1. This led to numerous debugging incidences after failing their test of the robot. In Lesson 4, students performed Task 2, a more complex task where the robot were developed to traverse volcanic areas to stop and collect three samples. Their conversations showed increased algorithmic thinking (51) and decreased debugging (23). Their pattern recognition (17), decomposition (14), and prediction (10) increased significantly from Lesson 3. In Lesson 5, after reviewing an additional mathematical concept ($\text{speed} * \text{time} = \text{distance}$), students continued to exercise algorithmic thinking. We anticipated that students would recognize more patterns using their experiences from Lessons 3 and 4. However, pattern recognition (13) was evidenced only in Pair 1's problem-solving process. Another skill that was exercised more was prediction, which was evidenced in the conversations of the more interactive pairs (1 and 2).

Table 2. The frequencies of CT sub-skills exercised during the collaborative problem-solving process

CT Sub-Skills	Pair 1			Pair 2			Pair 3			Across three pairs			Total
	3	4	5	3	4	5	3	4	5	3	4	5	
Lesson	3	4	5	3	4	5	3	4	5	3	4	5	
# of Events	16	10	8	12	14	10	9	6	11	37	30	29	96
# of Interactions within an Event	16	10	8	12	14	10	7	4	8	35	28	26	89
Abstraction	16	10	8	12	14	10	7	4	8	35	28	26	89
Debugging	13	9	7	10	11	12	6	3	6	29	23	25	77
Algorithmic Thinking	18	26	24	12	15	17	6	10	3	36	51	44	131
Decomposition	5	6	8	1	3	2	2	5	4	8	14	14	36
Pattern Recognition	2	6	13	1	5	3	1	6	3	4	17	19	40
Prediction	1	7	11	1	2	4	0	1	0	2	10	15	27
Total	190			135			75						400

Trajectories of CT Skills Exercised Over the Course of Collaborative Problem-Solving Process Between Pairs

Pair 1

Ashley and Bobby were able to complete Task 1 at the end of Lesson 3 and Task 2 at the end of Lesson 5. Three themes emerged from their collaborations that deserve a closer examination. First, Pair 1 was the most collaborative in that their conversations contained more CT skills (190) than the other two pairs, 75 and 35, respectively (see Table 2).

Second, with their continued demonstration of collaborative problem-solving, there were more pattern recognition (from two in Lesson 3 to 13 in Lesson 5), and prediction (from one in Lesson 3 to eleven in Lesson 5), and fewer debugging exercises (from thirteen in Lesson 3 to seven in Lesson

5) as the task became increasingly complex. Their conversations showed that they helped each other recognize the similarities between Task 2 and Task 1, and then they utilized the important principle (e.g., speed * time = distance) to perform the ensuing task. They also exercised more prediction by moving their body to simulate the movement of the robot as they read the codes on a computer screen. This enabled them to program the robot more truthfully to their expectations, resulting in less debugging.

Third, their CT capacities progressively reached an advanced level as they utilized these capacities more efficiently in face of the complex task. For example, their decomposition ability was enhanced although the frequency of decomposing practices was about the same across three lessons. In Lesson 3, Pair 1 decomposed a relatively

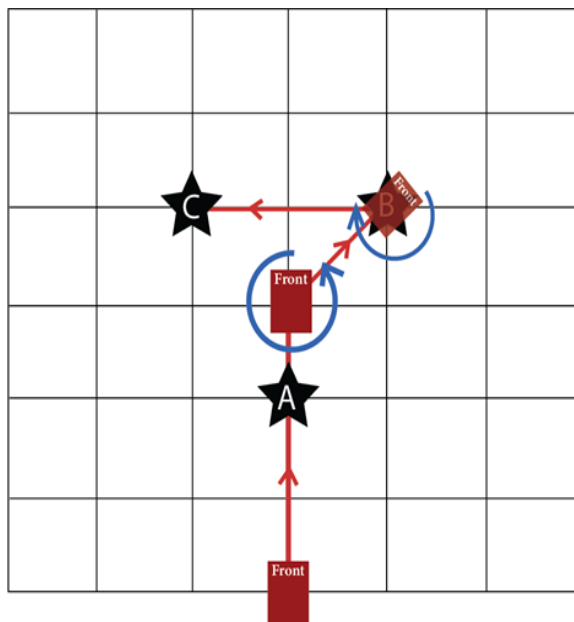
simple task into two steps, with the first step being “moving forward for one square” and second step being “making a 90 degree turn.” However, when tasked to perform the more complex task in Lessons 5, they were able to decompose the task at a larger granularity level. That is, they treated “moving forward for two squares, turning left, and moving forward for one square” as one step when the robot had to repeat a similar series of moves several times to accomplish the task.

Pair 2

Cameron and David actively collaborated throughout the lessons. They successfully completed Task 1. During Lesson 4, although they were the only pair that could drive the robot to pass three sample sites, they missed making the robot stop at each site. Thus, in Lesson 5, they designed the robot path and programmed the robot all over again. They

were able to successfully stop and collect three samples at the end of Lesson 5. When solving Task 1, Pair 2, like the other two pairs, engaged in numerous debugging activities with a trial-and-error approach. In Lesson 4 and 5, they continued to demonstrate a high frequency of debugging practices, 11 and 12. Most of their debugging behaviors resulted from their attempts to move the robot diagonally (see Figure 2) in order to save travel time, rather than along the grid lines with right angle turns. Such an approach took algorithmic thinking (15 times in Lesson 4 and 17 in Lesson 5) to calculate the time needed for making acute angle turns. Consequently, they engaged in many rounds of debugging. With their concentration mainly on figuring out the time needed for the robot to turn at the desired angles, they continued a trial-and-error approach and did not decompose and recognize the patterns as frequently as the other two pairs.

Figure 2. Pair 2 Designed route



Their strategy for solving the problem was focused on figuring out the solution for the subsets of the problem at the micro level, such as calculating the time needed for making desired turns and finding ways to make the robot stop when reaching the sample. Although they were collaborative in completing the task in Lesson 5, they rarely examined their earlier problem-solving approach. They infrequently used the previous task to help them identify patterns (3), reflect on their knowledge in order to decompose the task (2), and make predictions based on the program they have developed (4).

Pair 3

Eric and Fred demonstrated the lowest frequency of CT cognitive skills (75) in their collaboration, and they failed to complete Tasks 1 and 2. Overall, Pair 3 was the least collaborative, and talked with each other less frequently, especially in Lesson 5. At the beginning of Lesson 3, Pair 3 was relatively collaborative by exercising debugging (6) and algorithmic thinking (6). Like the others, they adopted a trial-and-error approach to move the robot to the designated location. Unfortunately, their robot failed to read the codes properly due to a technical issue. After their teacher fixed the robot, Fred started to disengage from the collaboration with Eric.

In Lesson 4, although Pair 3 encountered a technical issue again, they were more collaborative in solving the problem and were able to collect two samples at the end of this lesson. In particular, Fred applied CT sub-skills, pattern recognition and algorithmic thinking more often to help Eric solve the problem. Their conversations demonstrated more efforts to draw inferences from Task 1 to not only recognize the patterns (6) and apply algorithmic thinking (10) to determine

the timing and speed setting for the robot to reach the sample sites, but also to decompose the task into smaller sets of problems (5). This was the only time they showed prediction (1). In Lesson 5, although Eric and Fred talked with each other, most of their interactions were not relevant to the problems, which prevented them from making progress. In all, their frequencies of undertaking CT practices (24) in Lesson 5 were much lower than the other two pairs (71 and 48, respectively). Due to their fruitless collaboration, they did not exercise any *predictions* and produced fewer exercises in *algorithmic thinking*, *decompositions*, and *pattern recognitions* in the problem-solving process.

Discussion

Overall Pattern of CT Skills Involved in the Collaborative Problem-Solving Process

CT-related literature unanimously discusses that CT involves several cognitive skills at the conceptual level (Grover & Pea, 2013; NRC, 2010; Sengupta et al., 2013; Shute et al., 2017; Wing, 2011). The empirical evidence obtained from this study corroborates these theoretical papers in that students exercised multiple CT sub-skills as they worked out a solution to the given problem. Additionally, we found that some skills were applied more than the other during students' problem-solving process. Among the six CT skills, algorithmic thinking was exercised the most frequently. The quick adoptions of algorithmic thinking from our students is similar to that of Grover and colleagues' study (2015). When students are provided with opportunities and instructions for applying programming rules and mathematical concepts, students can develop algorithmic thinking and apply it naturally to perform a task. Debugging is

another cognitive skill that is also frequently seen in students' problem-solving process. Debugging dominated the problem-solving process when students first tackled the problem. Due to a lack of relevant experience, students tended to rely on a trial-and-error approach to solve the problem, which resulted in many rounds of debugging. This result is consistent with previous studies conducted in a programming education context (Lye & Koh, 2014; Quaye & Dasuki, 2017). In contrast, prediction and pattern recognition were CT skills exercised relatively less frequently by students in our study. This finding is in line with the observation reported by Kwon and Cheon (2019), that only one out of seven participants in their study recognized patterns during their coding process. The results demonstrate that certain CT skills may take more time for students to develop, such as pattern recognition.

Our study provides an initial understanding of the interactions among the identified CT skills, in contrast to the current literature. Most existing empirical studies examined only one CT skill, such as algorithmic thinking in Grover and colleague's study (2015) and decomposition in Kwon and Cheon (2019)'s study. We were able to notice associations among these skills. For example, when debugging behaviors decreased, we observed an increase in decomposition, pattern recognition, and prediction. This finding sheds further light on how different CT skills co-exist and/or interrelate with each other in a problem-solving process. According to Atmatzidou and Demetriadis (2016), with more time and practices, students regardless of gender and age will eventually develop CT skills. The CT skills exercised by our young participants suggest that some CT skills, such as debugging and algorithmic thinking, are relatively easier for students to grasp and

apply, whereas other skills, such as pattern recognition, decomposition, and prediction, require more practices, accumulated experiences, and perhaps deliberate efforts to develop and orchestrate in their problem-solving process.

Trajectories of CT Skills Exercised During the Collaborative Problem-Solving Process Across Pairs

The results of our study showed that students' uses of CT sub-skills over the course of collaborative problem-solving process varied from pair to pair, even though existing studies revealed that teamwork can help students perform better and enjoy more when solving problems (Atmatzidou & Demetriadis, 2016; Shute, et al., 2017). With teamwork, Pair 1 and Pair 2 successfully completed both tasks, whereas Pair 3 did not complete the tasks during the given time. Even though Pair 3 failed to complete the tasks, when the two students interacted to perform the tasks in Lesson 4, they showed their use of some advanced CT by moving away from the trial-and-error approach. Overall, students were more likely to arrive at a solution through collaborative problem-solving. However, the trajectories to reach the solution among group differed. Regarding the interactions for the two more collaborative pairs, interestingly they produced different patterns in their use of CT sub-skills. They both started with a trial-and-error approach, resulting in many debugging instances in Lesson 3. However, their applications of debugging varied starting in Lesson 4, with Pair 1 showing fewer while Pair 2 exhibiting more debugging. In the meantime, use of decomposition, pattern recognition, and prediction started to rise in both pairs. However, in Lesson 5, their pattern of applying these three skills diverged, with Pair 2 exercising more debugging behaviors

whereas Pair 1 utilizing significantly more decompositions, pattern recognitions, and predictions, which led to a more efficient and automated problem-solving process.

Upon examining the trajectories across pairs, we learned not only the importance of providing students with collaborative learning but also the need for guiding students to be well-versed in different CT skills, particularly those that take time to develop. As Shute and colleagues (2017) have pointed out, many teachers may not be trained sufficiently to teach CT. Our study results suggest that teachers should consider differentially monitoring and guiding students in relation to the development of particular CT skills.

Limitations

The first limitation of this study is limited generalizability, as we examined the recorded conversations from three pairs of students. Second, qualitative analysis could not avoid the subjective bias in the interpretation of participants' verbalization of thoughts, although we put much effort in dealing with the validity threat of observation bias while guaranteeing the rigor and reliability of the research. Last, our investigation was based on the recorded conversations and participants' think-aloud practices. There could be a chance that their actual thought processes differed from what they verbally expressed.

Implications for Future Research

First, we are aware of the need to further validate the findings of this research with a large-sample mixed method research. By extending the sample size of our next inquiry, we hope to increase the generalizability of our findings and thereby extend to further the ongoing effort to integrate CT in STEM

curriculum. In addition, future research might consider providing intentional instruction supporting students to develop advanced CT skills, such as decomposition, pattern recognition, and prediction, and conduct a comparison study with no guidance to possibly further younger students' CT capacities. Second, the task design for future research can be improved by providing tasks simulating real-life situations to enable the investigation of abstractive thinking skills. When examining our data, we found that our participants, different from the participants in Zhao and Shute's study (2019), demonstrated abstractive thinking throughout their collaborative experience, part of the reason being that the given tasks were specific and well-structured. Thus, this made it easier for students to identify the overarching goal of the tasks and focus on the most relevant information. For future research and practice, researchers may want to offer real-world ill-defined tasks coupled with additional contextual information, such as temperatures and the actual distance among each sample site, to create the scenarios that require students to apply abstraction in CT (Hong & Choi, 2019).

Conclusion

Our study aimed to uncover how students' CT sub-skills were used when interacting with peers to accomplish problem-solving tasks that require the integration of STEM knowledge. The results obtained from selected pairs of students' interactions showed that pairs that are more collaborative demonstrated more CT practices. Additionally, the different aspects of CT practices changed as students acquired experiences in solving the problems, in collaborating with others, and in integrating knowledge from different content areas. Our findings from our in-depth examination of different elements of CT students displayed

during collaborative problem-solving process shed further light into our understanding of how the different CT cognitive skills interact to enable individuals to process their thinking with automaticity to generate efficient solutions.

References

- Atmztidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, 75(B), 661-670.
- Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning*, 1(2), 65-81.
- Bers, M. U. (2017). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge.
- Bower, M., Wood, L. N., Lai, J. W. M., Howe, C., & Lister, R. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education*, 42(3), 53-72.
- Caeli, E. N., & Bundsgaard, J. (2020). Computational thinking in compulsory education: A survey study on initiatives and conceptions. *Educational Technology Research and Development*, 68(1), 551-573.
- Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*, 95, 202-215.
- Chi, M. T. H. (1997). Quantifying qualitative analyses of verbal data: A practical guide. *Journal of the Learning Sciences*, 6(3), 271-315.
- Dohn, N. B. (2020). Students' interest in Scratch coding in lower secondary mathematics. *British Journal of Educational Technology*, 51(1), 71-83.
- Doleck, T., Bazelais, P., Lemay, D. J., Saxena,

-
- A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: Exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education*, 4(4), 355-369.
- Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42(1), 38-43.
- Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, 25(2), 199-237.
- Hong, Y.-C., & Choi, I. (2019). Relationship between student designers' reflective thinking and their design performance in bioengineering project: Exploring reflection patterns between high and low performers. *Educational Technology Research & Development*, 67(2), 337-360.
- Iiskala, T., Vauras, M., Lehtinen, E., & Salonen, P. (2011). Socially shared metacognition of dyads of pupils in collaborative mathematical problem-solving processes. *Learning and Instruction*, 21(3), 379-393.
- ISTE and CSTA (2011). Operational definition of computational thinking for K-12 education [PDF document]. Retrieved from <https://www.iste.org/explore/articleDetail?articleid=152>
- Kwon, K., & Cheon, J. (2019). Exploring problem decomposition and program development through block-based programs. *International Journal of Computer Science Education in Schools*, 3(1), 3-16
- Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51-61.
- Manlove, S., Lazonder, A. W., & de Jong, T. (2006). Regulative support for collaborative scientific inquiry learning. *Journal of Computer Assisted Learning*, 22, 87-98.
- National Research Council. (2010). Report of a workshop on the scope and nature of computational thinking. Washington, DC: National Academies Press.
- Quaye, A. M., & Dasuki, S. I. (2017). A Computational approach to learning programming using visual programming in a developing country University. In P. Rich, & C.B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 121-134). Springer.
- Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behavior*, 72, 678-691.
- Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, 18(2), 351-380.
- Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142-158.
- Weintrop, D., Behest, E., Horn, M., Horton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127-147.

- Wing, J. M. (2011). Research notebook: computational thinking-what and why? Retrieved from <https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why>
- Zhao, W., & Shute, V. J. (2019). Can playing a video game foster computational thinking skills?. *Computers & Education*, 141, 103633.
- Zhong, B., Wang, Q., & Chen, J. (2016). The impact of social factors on pair programming in a primary school. *Computers in Human Behavior*, 64, 423-431.

Declaration of Interest Statement

Authors declare that they have no conflicts of interest. This study has been approved by the IRB of the University of Georgia (IRB Approval #: 000001432) and informed consent was obtained from all participants in the study. The datasets used in this study are unavailable due to privacy and data protection concerns.