European Journal of STEM Education,
2019, 4(1), 07
ISSN: 2468-4368

Revisiting Second Graders' Robotics with an Understand/Use-Modify-Create (U^2MC) Strategy

Youngkyun Baek [1]*, Sasha Wang [1], Dazhi Yang [1], Yu-Hui Ching [1], Steve Swanson [1], Bhaskar Chittoori [1]

[1] *Boise State University*, USA

***Corresponding Author:** youngkyunbaek@boisestate.edu

Citation: Baek, Y., Wang, S., Yang, D., Ching, Y.-H., Swanson, S. and Chittoori, B. (2019). Revisiting Second Graders' Robotics with an Understand/Use-Modify-Create (U^2MC) Strategy. *European Journal of STEM Education*, 4(1), 07. https://doi.org/10.20897/ejsteme/5772

Published: May 11, 2019

ABSTRACT

This study, a sub-study of a National Science Foundation (NSF) funded research project, applies a modified strategy of the U^2MC for an eight-week afterschool robotics curriculum to promote upper elementary students' computational thinking in the second grade. Twenty-one students in second grade participated in a Life on Mars project which lasted for ten days with one class hour per day. They participated in activities learning coding concepts, basics of robotics, as well as exploring life on Mars. Most notably, the study found a significant increase in participants' computational thinking skills. In addition, participants came to understand basic robotics, including operation, composites, and codes. Implications for future research and robotics curriculum design are discussed in the presentation.

Keywords: robotics, computational thinking, Use-Modify-Create approach, coding, STEM+C

COMPUTATIONAL THINKING AT ELEMENTARY GRADES

As computing becomes a routine but vital skill for everyone in today's society, educators realize that computer knowledge and skills are indispensable for future success. However, due to the ever-changing characteristics of computer knowledge and skills, this perspective soon shifts focus from knowledge and skills (which quickly become obsolete) to the thinking processes that govern computing tasks. Understanding the principle thinking processes is the key factor that would allow today's students to thrive in a future that is heavily influenced by computing (Yadav et al., 2014). Computational thinking is a process very similar to that of how computers operate. Computing refers to the parallel process the human brain takes in processing information, which aligns similarly to how computers function. Therefore, computational thinking is a process where human cognition behaves similarly to a computer. It is defined as a mental process involving these activities under each corresponding concept (Wing, 2006; 2008). Computational thinking draws on the above-mentioned concepts such as logic, algorithms, decomposition, abstraction, pattern recognition and generalization, and evaluation. Each concept systematically and efficiently processes information and tasks, and are fundamental steps in computer science (Lee et al., 2011). This involves logic as a mechanism for prediction and analysis (Ruggieri, 2000), making steps and rules characterized by algorithms, breaking down the whole into parts i.e., decomposition, removing unnecessary details which can be called an abstraction, spotting and using similarities to recognize patterns and drawing generalizations, and lastly making judgments for evaluation (Barr and Stephenson, 2011).

Because computational thinking is considered a prerequisite skill for many endeavors of the 21st century (Barr and Stephenson, 2011; Committee for the Workshops on Computational Thinking, 2010; Johnson et al., 2014; Wing, 2008; Yadav et al., 2014), educators have been developing curricula which integrate specific computational thinking skills, such as logical and algorithmic thinking (Barr and Stephenson, 2011; Johnson et al., 2014; Sullivan and Heffernan, 2016; Tran, 2018; Voogt et al., 2015; Weintrop et al., 2016). The primary goal of teaching computational thinking in schools is to develop students' ability to solve problems and express the solution in the form of an algorithm that enables the use a computer and other tools to assist in the solution. This approach to teaching computational thinking in schools is supported by The Computer Science Teachers Association (CSTA, 2011), Committee for the Workshops on Computational Thinking (2010), The Royal Society (2012), and Hemmendinger (2010). The integration of computational thinking into K-12 subjects has been mostly implemented in computer-related subjects such as computing and robotics, and other STEM (Science, Technology, Engineering, and Math) areas. While there already exists an implementation of computational thinking in the computing curriculum as a separate subject in the UK (Royal Society, 2012), the need to link computational thinking to subjects other than computer science has been argued by Wing (2006), Hemmendinger (2010), Voogt et al. (2015) and Fathi and Hildreth (2017). Currently, this approach to integration is mostly at the secondary level (ET 2020 Working Group on Digital Skills and Competencies, 2016), but the literature indicates an increasing trend to introduce it in primary education (Dasgupta et al., 2017).

On the Computational Thinking Vocabulary and Progression Chart created by ISTE and CSTA (2011), nine computational thinking vocabulary words/sets are presented for grade levels from Pre-K to 12th grade. It is not surprising to see activities for grades PreK to second are listed across all computational thinking vocabulary words. Learning experiences for PreK to second grade in language arts, geometry, literature, computer science, science, and social studies are exemplified. Even though ISTE and CSTA recommend cross-curricular activities for learning computational thinking, most practices were implemented around STEM subjects.

Chen et al (2017) organized a robotics curriculum around specific topics that control basic movements, voice recognition, tactile sensors, and various other robotics-related skills and tasks. Furthermore, the researchers included several key computer science concepts such as algorithms, variables, conditionals, loops, serial execution, and multitasking. Their participants followed three steps: writing a program on a piece of paper, testing it on a virtual robot, and running the tested program on a physical robot. They found that the devised activities in the curriculum improved students' computational thinking, especially in the context of robotics programming. As in this case of Chen et al. (2017), robotics is assumed as an interdisciplinary subject which can be used to integrate and practice computational thinking as it incorporates computer science and STEM concepts, systems, and technologies (Shoop et al., 2016).

Children as young as four can learn computational thinking concepts and robotics with assembling and programming (Grover and Pea, 2013). Kazakoff, Sullivan, and Baratè (2013) found that children's sequencing skills improved significantly from pre- to post-test after participating in a one-week intensive robotics and programming workshop. Bers, Flannery, Kazakoff and Sullivan (2014) demonstrated that kindergartners were both interested in and able to learn many aspects of robotics, programming, and computational thinking with the *TangibleK* curriculum design incorporating robot construction, programming concepts and debugging. In an attempt to replace the old curriculum with new robotics activities, Elkin, Sullivan, and Bers (2014) implemented a robotics curriculum which they integrated within a social science unit in a mixed-age Montessori classroom. The curriculum, which emulates the qualities of Montessori manipulatives, offers a unique way for children to interface with new concepts and perspectives through a lens customized to their past learning experiences. This study is one example of using robotics uniquely as a tool to provide enriching learning experiences for children, while still achieving programming skills and developing computational thinking skills.

Dasgupta et al. (2017) provided evidence of kindergarten students' engagement with computational thinking through analysis of their work focused on pattern recognition. They report that pattern recognition in a single direction is a developmentally appropriate skill for these kindergarten students; however, pattern recognition in two directions, both horizontally and vertically, was not commonly seen. It is challenging to introduce computational thinking into young students' activities due to their limited advanced cognitive skills, their complex learning environment, and cross-subject curriculum.

Baratè, Ludovico, and Malchiodi (2017) organized music notation activities with Lego blocks for elementary students. Characteristics of blocks such as shape, color, and position over the board were reconfigured to support multiple and heterogeneous encodings of a music score. Each block was used to have musical meaning. As a result, they could create a suitable learning framework which can improve music skills and, at the same time, can convey computational thinking concepts. Sullivan and Bers (2016) demonstrated that as early as pre-kindergarten, children were able to master foundational concepts regarding programming a robot, and that children as young as seven years old were able to master concepts as complex as programming a robot using conditional statements. Their curriculum was organized around knowledge of robots, robot construction, and programming a robot with conditional statements. In the study with five to six-year-old children, Lieto et al (2017) found a significant effect of educational robotics on robot programming skills after 13 training sessions of 75 minutes each, while incorporating hands-on experiences. They reported a short-term beneficial effect of robotics as demonstrated in the previous studies of Kazakoff et al. (2013) and Sullivan and Bers (2016). That is, they found that children learned to wait and check robot's moves and goals before relying on their behavioral control. As these studies have shown, educators can incorporate computational thinking skills early on in a grade-school curriculum to maximize learning potential across the years. It is important to note that separating the teaching method from the curriculum in robotics practices can be challenging. This is especially true in elementary grades where developmentally-appropriate and cross-curricular practices are not fully defined and identified (Dasgupta et al., 2017).

This study took the first implementation that utilized project-based learning, integrating a science topic, Life on Mars, with programming a robot into a second grade classroom. The Life on Mars project, based on scientific inquiry, was centered on designing robots and testing them on a simulated Mars environment in fourth and fifth-grade classrooms. In this project, students learned how to incorporate related science and engineering concepts into the designing of robots using Mindstorms EV3 Legos. The first implementation of this curriculum demonstrated that participating students and teachers enjoyed the afterschool STEM+C robotics program and the curriculum fostered students' development of computational thinking and positive attitudes toward science (Yang et al., 2018). However, the project-based learning approach used to develop students' programming concepts and computational thinking skills did not adequately prepare students to connect their learned knowledge and skills when tackling the problems in the final competition (Ching et al., 2018). Thus, this study adopted a modified strategy of the Understand/Use-Modify-Create to identify its effects on second grader's computational thinking skills.

STRATEGIES FOR PROMOTING COMPUTATIONAL THINKING IN ELEMENTARY EDUCATION

There are many ways to integrate computing education into existing curriculum in elementary education, especially as teaching computational thinking skills has taken various forms in its practice. Accordingly, the resources to support this emerging area in school curriculum continue to grow and become ever-more diverse. However, there is no denial that computational thinking integration into students' activities has been frequent in the subjects of computing and STEM (Science, Technology, Engineering, and Math). Traditionally, computing and STEM were the main subjects which dealt with the computational thinking skills of students. Recently it has begun expanding to other subjects such as the arts, reading, and music. Furthermore, cross-curricular activities to teach computational thinking has become a significant trend in elementary education (Tran, 2018). Ways of teaching and learning computational thinking skills are diverse, ranging from linear progressive strategies where a teacher leads the educative process of teaching computing to project-based, explorative inquiries where a student leads the process of learning computational thinking skills. Research and subsequent practices showing strategies in teaching computational thinking skills are grouped into four categories and introduced in the following paragraphs.

Manipulating Embodied Objects

In the context of problem-solving strategies as Lye and Koh (2014), and Butler et al. (2015) used a guided strategy, *divide and conquer*, which is a top-down deconstruction of building a castle in Scratch. Another similar strategy applied by Jörg et al (2014) for the purpose of teaching fifth and sixth graders computational thinking in order to increase the proportion of minorities and, specifically women in computing, was a step by step choreographed character animation. Baratè et al. (2017) used bricks, properly placed over a building baseplate to represent music scores in their study, which fostered computational thinking in elementary school children through LEGO-based music notation.

In the study performed by Good et al (2008), students ages 12 to 13 were engaged and motivated toward learning computational thinking using an embodied interface. The students in their study could take on the role of a character and organize the characters' movements into sequences so that the recorded movements could be manipulated in ways that fostered computational thinking. The hands-on activities with manipulatives seem to promote the decomposition skill, one of the computational thinking skills, as manifested by the study of Li, Hu, and Wu (2016), arguing that drawing geometric figures among the three hands-on activities was helpful in learning when the participants felt bored. One of the contributions of Good et al. (2008) and Li et al. (2016) was their addition of motivation and engagement in promoting computational thinking. Their *Unplugged and Embodied* games, together with the board games of Tsarava, Moeller, & Ninaus (2018), are examples of materialized physical manipulatives for elementary students to play with algorithmic thinking, abstraction, pattern recognition, and decomposition. The U^{2}MC strategy in this study includes activities manipulating embodied objects, that is, participants used their arms, legs, and eyes to embody and predict the robot's actions before and after they wrote their codes to make robots perform what they want to do.

Reflecting on Mistakes

Coding and programming are the most widely acknowledged ways of teaching computational thinking (Bauer et al., 2015; Lye and Koh, 2014) and therefore Lye and Koh (2014) asked students to verbalize their thought process using think-aloud protocol while programming. The think aloud process has been proved useful in fostering computational thinking, information processing, scaffolding, and reflection activities.

Frequently what happens to students while they are doing computer work is that computers do not always execute tasks as instructed. This is also true for students learning computational thinking skills. Students are encouraged to reflect on why their plan did not work, where possible mistakes reside, and how they could best fix their mistakes in their problem-solving. Bers et al. (2014) incorporated a debugging/trouble-shooting idea in their *TangibleK* Robotics project designed to engage kindergarten children in learning computational thinking, robotics, programming, and problem-solving. Harrison and Hanebutte (2018) turned coding mistakes into a pedagogy to teach computational thinking. Instead of merely showing and explaining correct solutions, they presented code with logical errors to class. Their teaching strategy consists of three stages: Initial Instruction, Problem Posing, and Addressal. The difficulty level might be critical in this way of teaching computational thinking because they are apt to get frustrated if the task is beyond their capability or the mistakes are hard to find or fix. This strategy should not be used until the students are comfortable with basic knowledge and constructs of coding (Harrison and Hanebutte, 2018). They recommended using this technique together with other methods for better balance. In this study, reflecting on mistakes was integrated into the U^{2}MC strategy. Thus, when the robot did not perform what the code instructed, participants came back to their code and analyzed them for mistakes, corrected it, and tried the revised code again. They repeated this trial and error process until they succeeded.

Creating a Story and Narration

Chiazzese et al. (2017) added a narrative stage for elementary students at the beginning of the teaching process of computational thinking as suggested by Repenning et al. (2017). Based on problem-based learning, Repenning et al. (2017) developed three stages of teaching computational thinking: abstraction for problem formulation, automation for solution expression, and analysis for execution and evaluation. Considering the elementary students' age, a storyline stage with a narrative description of characters and events in a story was added to the three stages by Chiazzese et al. (2017). The adoption of a narrative approach has stimulated a positive perception of computer programming for children in the study of Chiazzese et al. (2017).

Another narration strategy was adopted by Faber, Ven, and Wierdsma (2017). Students first created a pen-and-paper model of their story, consisting of drawings indicating what happens in the story. This model acts as an abstraction of the story, highlighting the most critical aspects, which then guides the coding process. A story or narration can be introduced as a scaffold either in the beginning or middle of teaching computational thinking concepts as revealed in the study of Webb and Rosson (2013). The students in this study were asked to make changes to the story to solve problems using broadcasting in Scratch. Webb and Rosson (2013) observed that scaffolded examples with story modifications in Scratch could provide an effective way to convey computational thinking concepts and skills in a short amount of time, while still serving as a fun and engaging learning activity. This 'Creating a Story and narration' strategy was integrated into the U^{2}MC strategy. Thus, when a new task was given to participants, they were asked to write a short story on how their robots will perform the task and what should be done in sequence.

U^{2}MC (Understanding/Using-Modifying-Creating)

The third strategy for teaching computational thinking includes using a prototype or sample, modifying or elaborating on it, and then creating a final project. The work of Figueiredo and García-Peñalvo (2017) illustrates this strategy. They introduced '*Practice Map Design*' and '*Follow and Give Instructions*' in order to practice and promote the computational thinking skills of students. With map designing exercises, students develop their capacities in planning, designing and describing specific characteristics in a concrete situation. Students are asked to draw on a paper what a student (or a teacher describes) and reverse roles. This strategy can also be applied without any digital devices.

Conde et al. (2017) promoted computational thinking by using unplugged methods and employing robots as teachers as an engagement factor for the students. The children played a game in class using colored game tokens where they were asked to write down the steps they took to complete a figure with colored tokens. In their study on making music with Scratch to teach computational thinking, Ruthmann et al. (2010) used a tangible computing device which is a midi card and an approach identical with the UMC/U^{2}MC approach. Once the participants in their study understand basic note and sound generation in Scratch, they implemented synchronization, and then more musical, generative algorithms for creating and manipulating sequences of notes could be explored. In this study, participants were given a set of codes, explained what the codes do, and then watched the robot's performance. Once they understood the codes, the instructor asked where in the program students could modify

Table 1. Bebras Challenge Scoring Rubric

Difficulty	Correct	Incorrect	Unanswered
A	+6	-2	0
B	+9	-3	0
C	+12	-4	0

the code to change the robot's action. All the participants were given a similar task as the example program with which to write their codes.

As is evident in the literature, practices integrating computational thinking in elementary education are growing. More resources are provided, and various strategies are emerging. Strategies promoting elementary children's computational thinking are based on problem-based learning, project-based learning, and game-based learning (Hsu et al., 2018). A modified U^{2}MC strategy in this study was integrated with strategies of Manipulating Embodied Objects, Reflecting on Mistakes, and Creating a Story and Narration.

RESEARCH QUESTIONS

In this study, the U^{2}MC (Understand/Use-Modify-Create) strategy (Lee et al., 2011) toward programming a robot was applied in addition to the project-based learning approach. Questions to be answered are as follows:

1. Could robotics activities with the U^{2}MC strategy promote computational thinking skills of the participants in second grade?
2. How the participants in second grade change in their knowledge of robots after they applied the U^{2}MC strategy in their robot activities?

METHODS

Participants

The study participants consist of 21 students in second grade at a suburban elementary in Boise, ID. They are 13 boys and nine girls. Academically, the students' reading and math abilities are at a second grade level range from low to high. At the commencement of the study, participants had already spent several hours with code.org during their math in classes and therefore were not unfamiliar with the languages in coding.

Instruments

Bebras Challenge for pre- and post-tests

The idea of Bebras Challenge was born by Dagienė (2006) as an international initiative on informatics at schools. In this study, the Bebras Challenge was administered before and after the implementation. The pretest showed that participants could not finish all questions within given forty minutes. After the pretest, all items were reversed to avoid memorizing questions. This challenge examines students' logic and computational thinking skills through different types of problems. There are three levels of difficulty: A, B, and C. A-level is easier than B, which are intended to be easier than the C-level, which involves a set of problem-solving skills. The challenges contain six tasks with 45 minutes to finish, two A-levels, two B-levels, and two C-levels. The challenge question 1, "Shelf Sort", measures Algorithmic Thinking (AL), Decomposition (DE), Evaluation (EV), and Algorithms and Programming (AP) domains that students can compare the rules that Beatrix set of itself as an algorithm and data can take many structures such as pictures or numbers each with different values. Question 2, "Broken Window" measures Abstraction (AB) and EV skills, and AP domains that students can store many pieces of information to share common attributes. Question 3 "Tube System" measures AL and AP domains that students need to command the mouse to go down and keep changing the directions until arriving at the cheese. Question 4, "Bottles" measures AB and EV skills, and Data structures and Representations (DR) domains that students solve which bottles should be at the front before they disappear behind one of the other bottles. Question 5, "Car Trip" measures AL and DE skills, and AP domains that students need to command the car to arrive at school. Question 6 "Secret Recipe" measures AL and DE skills and DR domains that students need to find which ingredient has no label.

U^{2}MC activity

In this study, students followed the U^{2}MC strategy integrated with other three strategies. The participants' activity was guided by a teacher who had received training by the researchers. In the first step, students ran a program that controls a robot and watched the program execution and the connected robot's actions; this is a 'Use' activity. After that, the teacher explained what the program is and what it does. Students are expected to have 'Understanding' through the teacher's explanation. After becoming familiar with the sophistication of the program, they began to modify it based on their understanding. In this 'Modify' stage, the teacher gave students brief ideas where they could make modifications, after which students had time to modify and play the program. Participants used their arms, legs, and eyes to embody and predict the robot's actions after they modified the program. This allowed participants to develop a deeper understanding and have a modified program for their robots to execute a different action from the original program. When the robot did not perform what the modified code instructed, participants came back to their code and analyzed it for mistakes, made corrections, and tried the revised code again. They repeated this trial and error process until they succeeded. In the 'Create' stage, they developed their ideas for a new robot project and deployed their knowledge and skills. That is, when a new task was given to participants, they were asked to write a short story on how their robots would perform the task and what should be done in sequence.

STEM Mars Robot Activities

Twenty-two students participated in the classroom activities in STEM Mars Robot. This activity provided K-2 classroom exposure to computer programming concepts and explored Mars learning outcomes (Yang et al., 2018). There were eight lessons in total for ten days to teach Mars Robot, geared towards developing the students CT skills, such as developing their ability to make algorithms, and to write and debug code.

In the breakout lesson, Lesson 1: Mars and Robots, students learned and shared ideas about Mars and Robotics. During this lesson, students had discussions about why humans would want to go to Mars or, alternatively, send a robot to Mars. This discussion provided a foundation for Lesson 2: Watching a video on robots used in our daily life; wherein students discussed what comprises a robot and how robots and robotics can be leveraged on Mars to potentially discover life and water. Lesson 3: Building a Robot brought the pre-learning from the previous lessons into the students' hands, and provided them with the opportunity to build their own EV3 Lego robot. During this lesson, groups of five students followed the robot building directions with teacher guidance as appropriate. To account for students who were at different centers, an alternative activity was leveraged in this lesson, in which these students participated in creating a robot prototype using Legos or read an informational text about Mars. After successfully building a robot prototype, students learned how to control a robot in Lesson 4: Controlling the Robot. This included basic movements and leveraging sensors to detect objects potentially obstructing the path of the robot by using the U^{2}MC approach to controlling a robot.

After Lesson 4, students used the UMC approach toward coding throughout the lessons. In Lesson 5: Algorithm, Coding, and Debugging, teachers showed the Cup Stack Pack to students and students created algorithms and coding to match the Cup Stack Pack. Students debugged code when necessary with guidance and support provided throughout the lesson, while still ensuring enough time for students to explore the basics of coding using the cups. In Lesson 6 & 7: My Loopy Robotics Friends, students chose a program from My Robotic Friends. This lesson served as a reintroduction to loops. Students developed critical thinking skills by looking for patterns of repetition in the movements of classmates and determining how to simplify those repeated patterns using loops. They identified places where the same arrow repeats consecutively and they wrote the number of repeated arrows inside the circle, while crossing out the other arrows in the repeated arrows. Similar to Lesson 5, teachers provided guidance and support, while still giving sufficient time for students to explore and practice. Finally, in Lesson 8: Working Mars Robots, students wrote how they used the computational model to problem solve during the lessons, such as building the robots, designing a robot, completing My Robotic Friends, and My Loopy Robotic Friends or discussing the activities of the Working Mars Robots.

Interview

To find out how students were learning about robotics and its programming, an interview was performed with questions such as 'What did you learn today?', 'What was new to you?', 'Was it difficult?", and other immediate facilitating questions such as 'Could you tell me more in detail?", etc. The interview was conducted four times by trained graduate students majoring in education and counseling: at the beginning of the implementation, after the students assembled their robots, after Lesson 3, and after all robotics activities. For the interview, two students from each group were selected at random, resulting in a total of eight students that were interviewed individually. The first interview, which was not included in the qualitative analysis, focused on robots in general, their parts and their role. It was an orientation towards the coming interviews. The second interview was conducted after lesson 5 and focused on coding. The third interview was conducted after all activities and focused on coding. The interview was conducted in a very comfortable, informal manner and students were asked to share what they learned. After students responded to the questions, the interviewer probed further to get additional information or clarification.

Table 2. Scores of pre and post-test of Bebras Challenge

Students	Pre-test	Post-test	Increase
1	2	18	16
2	-8	10	18
3	-6	-5	1
4	2	18	16
5	10	10	0
6	6	14	8
7	-10	2	12
8	2	10	8
9	-7	10	17
10	10	10	0
11	2	-2	-4
12	-9	0	9
13	6	10	4
14	10	10	0
15	-10	-10	0
16	-10	2	12
17	14	26	12
18	22	38	16
19	-10	-2	8
20	8	14	6
21	-10	2	12
Sum	16	175	159
Mean	.761	8.333	8.143

Table 3. Paired samples t-test of pre- and post-tests

Test	Mean Difference	Std. Deviation	t	df	Sig.
Post – Pre	8.143	6.733	5.542	20	.000

Table 4. Pre and post test scores by CT items

Item # (Type & sequence)	CT skill & domain	Sum of pre-test	Sum of post-test	Increase
A1	AL, DE, EV and AP	17	52	35
A2	AB, EV and AP	29	74	45
B3	AL and AP	-7	24	31
B4	AB, EV and DE	-5	36	41
C5	AL, EV, DR and AP	-10	-11	-1
C6	AL, DE and AP	-8	0	8
Sum		16	175	159

RESULTS AND DISCUSSION

Quantitative Analysis: Because one student missed the pre-test, twenty-one students' data obtained from the Bebras Challenge was analyzed. The pre- and post-test scores of the Bebras Challenge are presented in Table 2.

Sixteen students out of twenty-one showed increased scores from the pre-test to the post-test. Only five students showed the same or decreased in scores. The mean of the pre-test is .761 while that of the post-test is 8.333. The mean of the Increase is 8.143.

The paired samples test procedure in SPSS is presented in Table 3. The difference between the pre- and post-test is significant (t=5.542, df=20, p <.01) (Table 3). The most notable differences were in question type A and B. Type C did not show as much difference as type A and B (Table 4). According to Table 4, the sum of the difference is more than 35 in type A and B.

Type C, which has a higher difficulty level, did not show much difference between pre- and post-tests. This smaller or less significant difference may be due to the difficulty level and DR (Data Structures and Representations) trait in question C5.

Qualitative Analysis: The first interview performed before the implementation was not included in the qualitative analysis because it was focused on robots in general. The other three interviews were focused on identifying any changes in student knowledge and attitudes. The second, third and fourth interviews conducted during the implementation were recorded on video. The dialogues in the videos were transcribed and the researchers extracted any meaningful dialogues from the transcriptions. The findings are summarized below.

1) A robot is a machine, has a chip, and accepts orders to move

Findings show that the participants were narrowing down their understanding about a robot from 'it is a machine' to 'it needs to be programmed using codes' when they were asked 'What do you think a robot is?'. Two students did not answer or were not sure about this question, and six students' answers stated that 'it is a machine' at first interview, but later five students answered stated that 'it is a machine' or 'has the microchip', etc. They understood a robot reacts to commands or instructions'. Below, a few dialogues related to this observation at the second and third interviews are presented.

Student A: Well that a robot, it is a machine…what if it doesn't need an engine to move 'cause sometimes you could just have it in the microchip to tell it how to move?

Student B: What if a robot had no brain, no microchip, no programing?

Student C: Well that a robot, it is a machine…what if it doesn't need an engine to move 'cause sometimes you could just have it in the microchip to tell it how to move?

Student D: Chips.

2) No brain! It should be programmed

Student A: You have to tell the robot what to do—

Student B: - using the robot language instead of human language.

Student C: Well, I didn't know that we can make some robots and we can move or control them. That was kind of cool.

Interviewer: Yeah, it's neat. You can program them to do whatever you want them to do, pretty much, huh?

Student D: Yes. That how come we need to program a robot, but it can't move by itself. Then I watched a video, and it's because that it doesn't have a brain like we do. It doesn't have a heart. Yeah.

Student E: I can send my order to my robot one by one or send all at once. It follows my words.

The above dialogues show that the participants think a robot is a machine with no brain, they have chips instead to hold programmed instructions. They did not mention the codes, but later they learned to code to give controls to robots.

3) A robot acts only when commands are given from outside.

At the beginning of the implementation, eight of the students who were interviewed knew that a robot is an automatic machine that works autonomously. However, they did not know it should be instructed or programmed to do something. However, when they were programming a robot, they all could easily understand that a robot does nothing unless instructed, and only does precisely what the program directs. Dialogues with one student are presented below, as a representative of all the other seven students' dialogues.

Interviewer: How do you make a robot move?

Students G: I say and select, "move forward!". Then robot moves forward.

Interviewer: What happens if you do not say or select any command? It moves?

Students G: No, it is not moving, it is frozen, waiting for my command.

Interviewer: So, do you think when you have delivered a wrong command? The robot can judge if it is wrong?

Students G: No, the robot just accepts the command and just do it.

4) We repeat 'Trial and Error'

When they were writing the code to make a robot move or turn, they wrote a series of codes which consisted of breaking down the steps needed to achieve a given goal. After they typed the codes in on the computer, the students transferred all of codes into the chipset of the robot. Students then placed the robot on the floor and tested to see if it was executing the code as expected. If the robot was not executing the commands as instructed, students would then have to use problem solving skills to isolate where mistakes were made, and how it could be fixed. After identifying the mistakes, students would return to their computers and correct the codes and then test it again. This was a cyclical process until success was achieved.

Interviewer: What if something goes wrong? What do you do?

Interviewee: We solve it again.

..

Interviewer: Okay, how do you solve if your robot is not working as you think?

Interviewee: A mistake and then we fix it again, or something like that.

..

Interviewee:	Okay. I would try and make as much mistakes ever to—so I write down the mistakes I make and so I have a perfect robot.
..	
Interviewer:	In what ways? Yeah, that's what I'm asking.
Interviewee:	Maybe, if there's a piece. If it doesn't go in that side, maybe you should try another side and put it in there.
Interviewer:	Yeah. You find a problem, then you research it and find different ways to solve it. Yeah. Rita? Do you want to say something about the chart?
Interviewee:	If we make a problem, then we can break it.
Interviewee:	Break it?
Interviewee:	If we're trying to make a robot and we make—go in, we can take it apart, and we'll know, and then we can research it, then fix it.

CONCLUSIONS

In this study, the U^{2}MC (Understand/Use-Modify-Create) strategy toward programming a robot was applied in addition to the project-based learning approach model. It was anticipated that U^{2}MC strategy could be relevant for students in second grade because it makes sure they understand computational steps entirely before using, modifying and creating their own programs. The participants in second grade performed robotics programming successfully using the U^{2}MC strategy. The 'Understand' activity before 'Use' is assumed to be effective in engaging participants and making the activity effective towards the end-goal of robotics programming.

The participants in this study showed increased knowledge about robots and computational thinking skills after the STEM+C robotics activities. Throughout the interviews, they began to understand that a robot is a machine, has a chip, and accepts orders to move; that it has no brain like a human being and should be programmed using commands given from outside; and that 'Trial and Error' happens when they program robots.

At the easy and intermediate challenge levels for computational thinking, students' increase in computational thinking skills proved to be greater than at the most difficult level of challenge. However, it is uncertain whether the immeasurable changes in student computational thinking at higher levels of difficulty could be a result of the domain of computational thinking, that is, Data Structure and Representation. To simplify, the participants might not be ready for this domain of computational thinking, or it may be due to the lesson not being modified to adequately teach the domains to second graders. In Chen et al.'s (2017) study, the students in fifth grade did not show significant gains on Data and Representation of computational thinking domains either.

The results of this study indicate that further study is needed to clarify the readiness of second graders toward this domain of computational thinking. Additionally, this study found that the STEM+C robotics activities have given the participants in second grade opportunities to understand the basic principles of robotics: operation, composites, and codes. Students demonstrated the understanding that a robot is a machine composed of several parts consisting of chips similar to a human brain. The chipset in each robot can accept instructions written in codes to execute tasks, and that erroneous codes can be corrected through systematic testing and debugging.

Further studies should be done to determine which domains of computational thinking can be mastered by students in second grade, exploring which coding concepts are more appropriate for this age bracket. Finally, difficulty levels in computational thinking skills and coding concepts should also be clarified further in subsequent studies.

ACKNOWLEDGEMENT

We wish to thank the National Science Foundation (NSF) for the support of this study under Grant Number 1640228. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal*, *55*(7). 832-835. https://doi.org/10.1093/comjnl/bxs074

Baratè, A., Ludovico, L. A. and Malchiodi, D. (2017). Fostering computational thinking in primary school through a LEGO-based music notation. *Procedia Computer Science*, *112*(2017), 1334–1344. https://doi.org/10.1016/j.procs.2017.08.018

Barr, V. and Stephenson, C. (2011). What is involved and what is the role of the computer science education community? *ACM Inroads*, *2*(1), 48-54. https://doi.org/10.1145/1929887.1929905

Bauer, A., Butler, E. and Popovic, Z. (2015). Approaches for teaching computational thinking strategies in an educational game: A position paper. In *Proceedings of the IEEE Blocks and Beyond Workshop (*121 – 123). https://doi.org/10.1109/BLOCKS.2015.7369019

Bers, M. U., Flannery, L., Kazakoff, E. R. and Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*(2014), 145-157. https://doi.org/10.1016/j.compedu.2013.10.020

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X. and Eltoukhy, M. (2017) Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers and Education*, *109*(C), 162-175. https://doi.org/10.1016/j.compedu.2017.03.001

Chiazzese, G., Fulantelli, G., Pipitone, V. and Taibi, D. (2017). Promoting computational thinking and creativeness in primary school children. In J. M. Dodero, M. S. Ibarra Sáiz, & I. Ruiz Rube (Eds.), *Fifth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'17),* October 18-20, 2017, Cádiz, Spain. (Article 6). New York, NY, USA: ACM. https://doi.org/10.1145/3144826.3145354

Ching, Y.-H., Yang, D., Wang, S., Baek, Y., Swanson, S. and Chittoori, B. (2018). Improving student attitudes in STEM through a project-based robotics program. *A paper presented at American Educational Research Association (AERA) Annual Meeting and Exhibition*, New York, NY, USA, April 13-17, 2018.

Committee for the Workshops on Computational Thinking; National Research Council (2010). *Report of a workshop on the scope and nature of computational thinking*, The National Academies Press. Washington, D.C.

Conde, M. Á., Fernandez-Llamas, C., Rodríguez-Sedano, F. J., Guerrero-Higueras, Á. M., Matellan-Olivera, V. And García-Peñalvo, F. J. (2017). Promoting computational thinking in K-12 students by applying unplugged methods and robotics. In J. M. Dodero, M. S. Ibarra Sáiz, & I. Ruiz Rube (Eds.), *Fifth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'17)*, October 18-20, 2017, Cádiz, Spain. *(Article 6).* New York, NY, USA: ACM. https://doi.org/10.1145/3144826.3145354

Dagienė, V. (2006). Information technology contests - introduction to computer science in an attractive way. *Informatics in Education*, *5*(1), 37 – 46.

Dasgupta, A., Rynearson, A. M., Purzer, S., Ehsan, H. and Cardella, M. E. (2017). Computational thinking in K-2 Classrooms: Evidence from student artifacts (Fundamental). *A paper presented at American Society for Engineering Education Annual Conference and Exhibition*, June 25-28, 2017, Columbus, Ohio. https://doi.org/10.18260/1-2--28062

Elkin, M., Sullivan, A. and Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom. *Journal of Information Technology Education: Innovations in Practice*, *13*, 153-169. https://doi.org/10.28945/2094

ET 2020 Working Group on Digital Skills and Competences (2016). *Coding and computational thinking on the curriculum, Key messages of PLA#2.* Retrieved from https://ec.europa.eu/education/sites/education/files/2016-pla-coding-computational-thinking_en.pdf

Faber, H. H., Ven, J. S. and Wierdsma, M. D. (2017). Teaching computational thinking to 8-year-olds through ScratchJr. *ITiCSE '17 Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 359-359). July 03-05, 2017, Bologna, Italy. https://doi.org/10.1145/3059009.3072986

Fathi, R. and Hildreth, D. (2017). Computational thinking and application – a cross curricular approach to teach computer science in high school chemistry classrooms. *2017 Proceedings of the EDSIG Conference*, v3 n4357, Austin, Texas USA

Figueiredo, J. and García-Peñalvo, F. J. (2017). Improving computational thinking using follow and give instructions. In J. M. Dodero, M. S. Ibarra Sáiz, & I. Ruiz Rube (Eds.), *Fifth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'17)*, October 18-20, 2017, Cádiz, Spain. (*Article 3*). New York, NY, USA: ACM. https://doi.org/10.1145/3144826.3145351

Good, J., Romero, P., du Boulay, B., Reid, H., Howland, K. and Robertson, J. (2008). An embodied interface for teaching computational thinking, *Proceedings of the 13th international conference on Intelligent user interfaces (pp. 333-336)*, January 13-16, 2008, Gran Canaria, Spain. https://doi.org/10.1145/1378773.1378823

Gouws, L., Bradshaw, K. and Wentworth, P. (2013). Computational thinking in educational activities: An evaluation of the educational game Light-Bot. *ITiCSE'13*, July 1–3, 2013, Canterbury, England, UK. https://doi.org/10.1145/2462476.2466518

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

Harrison, W. S. and Hanebutte, N. (2018). Embracing coding mistakes to teach computational thinking. *Journal of Computing Sciences in Colleges*, *33*(6). 52-62.

Hemmendinger, D. (2010). A please for modesty. *ACM Inroads*, *1*(2), 4–7. https://doi.org/10.1145/1805724.1805725

Hsu, T. C., Chang, S. C. and Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*(2018), 296-310. https://doi.org/10.1016/j.compedu.2018.07.004

International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) (2011). *Computational thinking teacher resources*, second edition. Retrieved from https://www.csteachers.org/

Johnson, D., Black, B., McGriff, M. and Daney, K. (2014). Using technology to promote computational thinking in middle school classrooms. In M. Searson & M. Ochoa (Eds.), *Proceedings of Society for Information Technology & Teacher Education International Conference 2014* (pp. 2866-2868). Chesapeake, VA: Association for the Advancement of Computing in Education (AACE).

Jörg, S., Leonard, A. E., Badu, S., Gundersen, K., Parmar, D., Boggs, K. and Daily, S. B. (2014). Character animation and embodiment in teaching computational thinking. *SIGGRAPH '14 ACM Posters Article No. 4*, August 10 – 14, 2014, Vancouver, British Columbia, Canada. https://doi.org/10.1145/2614217.2630597

Kazakoff, E., Sullivan, A. and Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal*, *41*(4), 245–255. https://doi.org/10.1007/s10643-012-0554-5

Kazimoglu, C., Kiernan, M., Bacon, L. and MacKinnon, L. (2012). Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, *9*, 522-531. https://doi.org/10.1016/j.procs.2012.04.056

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. and Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32-37. https://doi.org/10.1145/1929887.1929902

Li, W. L., Hu, C. F. and Wu, C. C. (2016). Teaching high school students computational thinking with hands-on activities. *ITiCSE 2016: Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education* (pp. 371-371). July 11-13, 2016. Arequipa, Peru. https://doi.org/10.1145/2899415.2925496

Lieto, M. C. D., Inguaggiato, E., Castro, E., Cecchi, F., Cioni, G., Dell'Omo, M., Laschi, C., Pecini, C., Santerini, G., Sgandurra, G. and Dario, P. (2017). Educational robotics intervention on executive functions in preschool children: A pilot study. *Computers in Human Behavior*, *71*(2017), 16-23. https://doi.org/10.1016/j.chb.2017.01.018

Liu, H. P., Perera, S. M. and Klein, J. W. (2017). Using model-based learning to promote computational thinking education. In P. J. Rich and C. B. Hodges (Eds.) *Emerging research, practice, and policy on computational thinking* (pp. 153-172). Cham, Switzerland: Springer International Publishing AG. https://doi.org/10.1007/978-3-319-52691-1_10

Lye, S. Y. and 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*(C), 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Repenning, A., Basawapatna, A. R. and Escherle, N. A. (2017). Principles of computational thinking tools. In P. J. Rich & C. B. Hodges (Eds.) *Emerging research, practice, and policy on computational thinking* (pp. 291-305). Cham, Switzerland: Springer International Publishing AG. https://doi.org/10.1007/978-3-319-52691-1_18

Ruggieri, S. (2000). Decidability of logic program semantics and applications to testing. *The Journal of Logic Programming*, *46*(1-2). 103-137. https://doi.org/10.1016/S0743-1066(99)00067-9

Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P. and Saulters, C. (2010). Teaching computational thinking through musical live coding in Scratch, *SIGCSE'10 - Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 351-355). 41st ACM Technical Symposium on Computer Science Education, Mar 10-13, 2010. Milwaukee, WI, United States. https://doi.org/10.1145/1734263.1734384

Shoop, R., Flot, J., Friez, T., Schunn, C. and Witherspoon, E. (2016). Can computational thinking practices be taught in robotics classrooms? *Presented at the International Technology and Engineering Education Conference*, National Harbor, Washington DC. March 2-4, 2016.

Sullivan, F. R. and Heffernan, J. (2016) Robotic construction kits as computational manipulatives for learning in the STEM disciplines. *Journal of Research on Technology in Education*, *48*(2), 105-128, https://doi.org/10.1080/15391523.2016.1146563

The Computer Science Teachers Association (CSTA, 2011). *CSTA K-12 Computer science standards, Revised edition.* CSTA, New York.

The Royal Society (2012). *Shut down or restart? The way forward for computing in UK schools.* London: The Royal Society. Retrieved from https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

Tran, Y. (2018). Computational thinking equity in elementary classrooms: What third-grade students know and can do. *Journal of Educational Computing Research*, January, 2018. https://doi.org/10.1177/0735633117743918

Tsarava, K., Moeller, K. and Ninaus, M. (2018). Training computational thinking through board games: The case of Crabs & Turtles. *International Journal of Serious Games*, *5*(2), 25-44. https://doi.org/10.17083/ijsg.v5i2.248

Voogt, J., Fisser, P., Good, J., Mishra, P. and Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, *20*(4). 715–728. https://doi.org/10.1007/s10639-015-9412-6

Webb, H. C. and Rosson, M. B. (2013). Using scaffolded examples to teach computational thinking concepts. *SIGCSE '13 Proceeding of the 44th ACM technical symposium on Computer science education*, (pp. 95-100). March 06-09, 2013. Denver, Colorado, USA. https://doi.org/10.1145/2445196.2445227

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L. and Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, *25*(1), 127-147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–36. https://doi.org/10.1145/1118178.1118215

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences*, *366*(1881), 3717–3725. http://doi.org/10.1098/rsta.2008.0118

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S. and Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education*, *14*(1), *Article 5*, 1-16. https://doi.org/10.1145/2576872

Yang, D., Baek, Y., Ching, Y., Swanson, S., Chittoori, B. and Wang, S. (2018). The design and enactment of a computational thinking-rich project-based integrated K-12 STEM curriculum. *Manuscript submitted for publication.*