WearableLearning: Developing Computational Thinking through Modeling, Simulation and Computational Problem Solving

Injila Rasul, Danielle Crabtree, Francisco Castro, Allison Poh, Sai Satish Gattupalli, Krishna Chaitanya Rao Kathala, Ivon Arroyo
irasul@umass.edu, dcrabtree@umass.edu, fcastro@cs.umass.edu, apoh@umass.edu, sgattupalli@umass.edu, kkathala@umass.edu, ivon@cs.umass.edu
University of Massachusetts Amherst

Abstract: Computational Thinking (CT) is a vital and multi-dimensional skill for all 21st Century Learners. In this study, we investigated the development of three aspects of CT: Self-Perception of Computational Ability, Modeling and Simulation, and Computational Problem Solving, as students engaged in collaborative game design and programming practices. This study contributes evidence for the development of two of these CT dimensions, Modeling and Simulation and Computational Problem Solving, through their engagement with the WL curriculum and platform. We found increases in students' ability to understand machines and their processes, alongside an improved capacity to think algorithmically as they constructed models, debugged, and iterated through their designs.

Introduction

Computational thinking (CT) has been acknowledged by educators as an essential skill to master within STEM fields (Wing, 2006). Considered a key aspect of Computer Science (Grover & Pea, 2018), CT is now understood to more broadly encompass the thought processes and practices necessary for systematic problem-solving, such as iterative testing, algorithmic thinking, and troubleshooting (Weintrop et al., 2016; Shute et al, 2017; Resnick et al., 2009). In schools, CT is not typically included within the K-12 curriculum in an actionable way, therefore, this research focuses on demonstrating a possible program that addresses this gap within the STEM curriculum.

The WearableLearning Platform (WL) aims to develop students' CT skills, by having K-12 students and teachers create, administer and play multiplayer games, using mobile devices (Arroyo et al., 2022). This study focused on understanding the impact of the WL curriculum that involves gameplay, game design, programming, and gameplay testing. The research question that we explore is: what aspects of Computational Thinking does the process of game playing, game ideation, and programming/implementation impact? We hypothesized that the aforementioned process would yield an increase in students' CT abilities as understood by a multi-faceted conceptualization of CT, in particular: how they see their own abilities to solve problems, their understanding of machines and models that represent their functioning, and their capability to generate correct solutions that use logical reasoning.

Literature Review

Computational Thinking, coined by Wing (2006), has grown to become an umbrella term that refers to a broad set of early Computer Science skills, which are essential to thrive in our increasingly digital world. Many scholars have grappled with the question "what is CT" and have contributed different, highly context-driven answers. Most definitions include aspects of problem decomposition, pattern recognition, data representation, generalization and abstraction, systems thinking, and algorithm-building (Grover & Pea, 2018). Another one of the most thorough analyses of CT at the K-12 level (Weintrop et al., 2016) suggested the following practices: analyzing and logically organizing data; data modeling, abstractions, and simulations; formulating problems so computers may assist; identifying, testing, and implementing possible solutions; automating solutions via algorithmic thinking; and generalizing this process to other problems. We agree with the reflection by Román-González et al. (2019), which suggests that the Computational Thinking term has helped to extend Computer Science Education beyond computer programming, and helped to lower the barriers to entry for computer programming, in part due to an increase in the number of visual block programming languages. In addition to cognitive skills, engaging in computational thinking offers the opportunity to develop noncognitive aspects related to attitudes and 21st-century skills such as persistence, self-confidence, tolerance to ambiguity, creativity, and teamwork (van Laar et al., 2017). Thus, the use and application of the CT term has evolved and grown beyond strictly CS education (Kalelioglu et al., 2016).

CT has allowed us to frame underlying mental processes that happen during tasks associated with programming, enhancing the metaphor of "programming to learn" instead of "learning to program". We believe that the computational thinking process is at the intersection of several cyclical and interconnected processes,

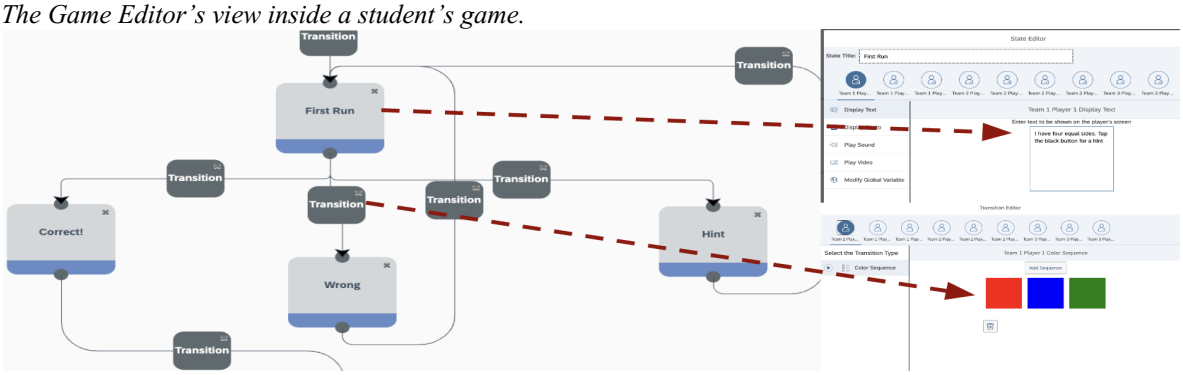
including the engineering design process (Ertas & Jones, 1996) (from an engineering perspective), problem solving (Polya, 1945) (from a mathematics perspective), and the iterative design process (Nielsen, 1993) (from an HCI perspective). The difference being that the final product in CT is a logic-based computational solution to a problem, defined at various possible levels of abstraction. In all cases, the kind of thinking that CT encompasses, is a cyclical process where a problem is explored at a high level of detail and precision, with multiple possible solutions that need to be articulated and implemented to varying degrees, tested and evaluated according to a criterion of success, and revised and redefined.

Overview of the WL Platform

The WL Platform is an online platform through which users can create and deploy active math games without prior programming experience. The programming interface (see Figure 1) consists of States and Transitions that the students can drag and drop to connect and create a game flow. The States contain the specifics of the display screen contents for each screen that the user sees and can include text, photos, sound, videos, and global variables. How one State moves onto the next depends on the conditional Transition placed on their connection. Transitions can be text entries, color codes, or customized button choices. The Game Creator mode has a Run & Debug functionality, which runs an iteration of their code and sees what the Game Player would on their screen as they progress through the game, allowing them to troubleshoot, debug and fix errors as they code.

Figure 1

The Game Editor's view inside a student's game.



Overview of the WL Curriculum

The WL curriculum consists of a series of guidelines and materials for 8 class sessions, to support teachers in guiding their students through a problem-solving process (see Table 1). The overarching goal is to design a mathematical game, with progress and collaboration aided by mobile devices, for others to play and learn or practice math skills. Based on our past studies for this project, playing an existing math game on the WL platform first helped students understand the possibility of ideating multiplayer math games that easily integrate technology, therefore students play existing games from the WL library, before creating new ones.

Students work through an iterative design process, engaging in designing, prototyping, and testing/evaluating their game designs, and redesigning after feedback from other students. Throughout, students engage with a 5-stage engineering design process specific to WL (Table 1): Game Playing; Brainstorm Game Design; Develop Finite State Machine Diagrams; Implement and Debug; and Exchange and Play Games.

Methodology

The study was conducted with a total of 47 students (ages 11–13), across two after-school programs in Eastern and Western Massachusetts, over 8-10 hours of contact time. Across both programs, 46.8% of students self-identified as male, 38.3% self-identified as female, and 14.9% preferred not to answer.

Instruments

The pre- and post- tests measured three different aspects of CT: Self-Perception of Ability, Modeling and Simulation, and Computational Problem Solving.

The first section of the test gleaned their self-concept of how adept they considered themselves at computational tasks and the use of computers, adapted from the work on CT perspectives by Angeli et al. (2016). Angeli and colleagues (2016) frame important notions of what students at different grades should know to have

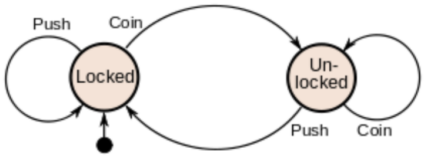
Computational Thinking. We created a test of CT self-concept by adapting these notions to feelings of knowing CT skills. Seven questions were asked on a 5-point Likert scale (Strongly Agree to Strongly Disagree).

The second part of the test posed questions related to modeling a machine (see Figure 2 for sample items on Finite State Machines assessment). According to Wilensky et al. (2016) Computational Thinking activities in various capacities include learners designing, constructing, and evaluating models as part of their educational activities. We decided to assess the modeling and simulation ability of students, by assessing their ability to interpret abstract models of finite-state-automata, what kinds of machines they represent, how they function, especially given the heavy reliance of our curriculum and programming language on Finite State Machines, for students to program and specify the behaviors of the mobile devices in their games, as assistants to the players. We created items on Finite State Automata from beginner material given to students in College as part of Computer Science classes.

Figure 2

The three items that assessed students' knowledge of Finite State Automata

This is a visual model of how a machine works.



1. Can you tell what kind of machine this could be? What does it remind you of and why?
2. According to this model, what would happen if you put three (3) coins in?
3. The machine is currently locked. What would you have to do to get this machine to go "unlocked" and then back to "locked"?

The final part of the test had items from the CTt (Roman Gonzalez et al., 2019) relevant to algorithmic thinking concepts such as sequencing and loops, to measure computational problem-solving skills. According to Wilensky (2016), this involves a variety of skills, from preparing problems for computational solutions, choosing tools, assessing approaches/solutions to a problem, debugging, etc. The Computational Thinking test addresses some of these aspects and was adapted from the CTt (Roman-Gonzalez et al., 2019), and shortened to only include the preliminary concepts of algorithmic thinking, excluding the more complex nested loops and conditionals items, as those were not part of the curriculum. The items were multiple-choice.

Results

A total score was computed for each student for the three pretest/posttests, with the score for self-perception being scored as an average of the Likert scale scores for each student, and the FSM and CTT components being sums to indicate correctness. Descriptive statistics were computed for the collected data, and the difference between pre and post-test scores was analyzed using a two-tailed paired-samples t-test comparison. Significance was compared to an alpha level of 0.05 (see Table 2).

We found that there was a statistical significance in the CT measures for Modeling & Simulation ($t = 2.01$, $p = 0.05$) and Computational Problem Solving ($t = 2.62$, $p = 0.01$), from pre to post-test. Modeling & Simulation showed a small to medium effect size, indicating that exposure to the study positively impacted students' skills of creating models and understanding of Finite State Machine concepts. The Computational Problem Solving section showed a small to medium effect size, indicating the students' engagement with the content and process positively impacted their problem-solving skills.

We found no statistical significance for average scores on Self-Perception ($t=.74$, $p=.46$). This indicated that there was no pattern in the change of students' average self-perception over the course of the intervention.

Discussion and Conclusion

Students showed improvement in two (out of three) different measurements of Computational Thinking after being exposed to the WL curriculum and platform for 8-10 hours of contact time. They improved in Algorithmic Thinking/Computational Problem Solving Practices and Modeling/Simulation practices, which were reflected in their work as well as they developed complex models for their games and debugged those models as well.

Students did not improve in their Self-Perceptions of their Computational Thinking abilities, which probed students to think about the level of comfort and ease with which they can engage in computational practices and use digital media to problem-solve. There are multiple possibilities why there was no significant improvement in their self-perceptions of their CT abilities. This might be because of the steep learning curve experienced during the 8-hour curriculum, or because it was their first experience as an active creator of technology instead of being passive users of technology. This experience might have challenged their self-perception of their

technology/digital skills. These findings reflect similar results in active vs. passive STEM learning that suggest novice learners are inaccurate in their judgments of how much they learned, and this "feeling of learning" can be negatively correlated with actual learning (Deslauriersa et al., 2019).

Possible improvements may include extending the curriculum to 12-13 contact hours, which could make students more comfortable with the learning curve. Another possibility would be to add specific reflection prompts that encourage thinking about what skills they have developed as they participate in this process of game design and creation, thereby deliberately building their awareness of the skills they are developing.

The results of our study illustrate how different conceptualizations of CT are broad enough that an intervention targeting CT might influence some aspects of CT, yet not others. This finding opens the discussion for a more granular analysis of activities and their impact. We learned that different measurements/instruments of Computational Thinking, according to different authors and perspectives, could yield different results, as they capture only a portion of the conceptualization of CT. Thus, we recommend that other researchers should take a multidimensional framework as a way to measure CT, as it should yield a more accurate portrayal of CT ability in students and allow for more targeted interventions.

References

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. Educational Technology & Society, 19 (3), 47–57.

Arroyo, I., Closser, A. H., Castro, F., Smith, H., Ottmar, E., & Micciolo, M. (2022). The WearableLearning Platform: A Computational Thinking Tool Supporting Game Design and Active Play. Technology, Knowledge and Learning. https://doi.org/10.1007/s10758-022-09601-1

Deslauriers, L., McCarty, L. S., Miller, K., Callaghan, K., & Kestin, G. (2019). Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom. PNAS, 116(39), 19251–19257. https://www.pnas.org/doi/full/10.1073/pnas.1821936116

Ertas, A. & Jones, J.C. (1996). The Engineering Design Process. New York: John Wiley and Sons.

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. *Computer science education: Perspectives on teaching and learning in school*, *19*(1), 19-38.

Kalelioglu, F., & Gulbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. Baltic Journal of Modern Computing, 4, 583-596.

Nielsen, J. (1993). Iterative user interface design. IEEE Computer, 26(11), 32-41.

Polya, G. (1945). How to Solve It: A new aspect of mathematical method. Princeton, NJ: Princeton Univ. Press.

Resnick, M., Maloney, J., Hernández, An., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for Everyone. Communications of the ACM. 52. 60-67.

Román-González, M., Moreno-León, J., & Robles, G. (2019). Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions. In Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions, 79–98. https://link.springer.com/chapter/10.1007%2F978-981-13-6528-7_6

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

van Laar, E., van Deursen, A., van Dijk, J., de Haan, J. (2017) The relation between 21st-century skills and digital skills: A systematic literature review, Computers in Human Behavior, 72, 577-588. https://doi.org/10.1016/j.chb.2017.03.010.

Weintrop, D., Beheshti, E., Horn, M., Orton, 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. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2006). Computational thinking. Association for Computing Machinery, 49(3), 33–35. https://doi.org/10.1145/1118178.1118215

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants #2026722 and #2041785. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.