A DSML for a robotics environment to support synergistic learning of CT and geometry

Nicole Hutchins[1], Timothy Darrah[1], Hamid Zare[1], Gautam Biswas[1]
[1]Institute for Software Integrated Systems, Vanderbilt University
Nashville, TN USA
nicole.m.hutchins@vanderbilt,edu, timothy.s.darrah@vanderbilt.edu, hamid.zare@vanderbilt.edu, gautam.biswas@vanderbilt.edu

ABSTRACT
Synergistic learning of computational thinking (CT) and STEM has proven to be an effective method for enhancing CT education as well as advancing learning in many STEM domains. Domain Specific Modeling Languages (DSML) facilitate the building of computational modeling frameworks that are directly linked to STEM content, thus making it easier for students to focus on concepts and practices. At the same time, teachers can more easily relate curricular content to the model building tasks. This paper discusses the design, development, and implementation of a robotics DSML to support a middle school geometry curriculum.

KEYWORDS
DSML, robotics, STEM, geometry

1. INTRODUCTION
Recent developments show how computational tools have influenced research and practices in mathematics and science education (National Research Council, 2012). In parallel, rapidly evolving educational technologies have influenced pedagogy and curriculum development, primarily by integrating computational tools into the study of STEM disciplines (Grover & Pea, 2013, Hutchins, Zhang, & Biswas, 2017). While the limited availability of skilled teachers, financial constraints on educational institutions, and the inertia in changing current curricular practices has impeded the introduction of Computer Science (CS) courses into middle and high school classrooms, curricula supported by educational software that exploit the synergies between STEM and CT and integrate with current K-12 curricula have found success (Basu, Biswas, & Kinnebrew, 2017; Jona et. al., 2014, Sengupta, et. al., 2013; Weintrop, et. al., 2016).

In the past, model-based design has been employed to facilitate a necessary convergence among physical processes and software control design, thus supporting many Cyber Physical System (CPS) applications (Jackson & Sztipanovits, 2008; Jensen, Chang, & Lee, 2011). In this paper, we extend this design process to Open Ended Learning Environments (OELEs) and focus on the design and integration of curricular scaffolding in OELEs to support student learning in STEM and CS domains.

This paper outlines the development of a WebGME design studio centered on the application of a domain specific modeling language (DSML) for robotics to support a middle school mathematics curriculum. To do so, we analyze the literature and establish curricular and software requirements, describe the design and development of our WebGME design studio, and conclude with case studies from a usability study.

2. BACKGROUND
To implement a set of learning tasks, while assuring well-formed model realizations (Jackson & Sztipanovits, 2008), we conducted a thorough analysis on the DSML design requirements in combination with the curricular needs of a middle school mathematics classroom. Here we cover four topic areas that directly relate to our research.

2.1. Computational Thinking (CT)
Following Wing's call for the increased introduction of CT in classrooms (2006), significant work was completed towards an applicable definition as well as an outline of key concepts and practices that can be used to assess learning gains in CT. The Royal Society defined CT as "*the process of recognizing aspects of computation in the world that surrounds us and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes*" (Royal Society, 2012). In Grover and Pea's systematic review (2013), the authors listed essential CT constructs and, for the purposes of our work, we focused on flow of control, decomposition, efficiency and performance constraints, and debugging.

To facilitate CT and the acquisition of basic geometry skills, appropriate scaffolding must be incorporated into the design of the DSML. Significant success with synergistic learning of CT and STEM disciplines through the use of block-based DSMLs (Hasan & Biswas, 2017) has supported increased integration of this style of programming at the K-12 level and we seek to extend this effort through the use of a DSML created in a model-based design environment such as WebGME. In our platform, CT provides the framework for building computational models or algorithms to define and debug the movement of robots. The metamodel and model building visualizer described in Section 5 provide a level of curricular abstraction that eliminates many of the burdens of text-based programming. In addition, our model-based design environment is supported by a necessary utilization of CT constructs such as debugging and problem decomposition.

Furthermore, our robotics platform provides multiple representations with the utilization of a physical robot (as opposed to a virtual sprite), a physical coordinate plane, and a bird's eye view of the grid space with several overlays (e.g., movement traces, lines, points, etc). Abstraction is provided in the model building visualizer that the student uses to construct their command sequence. As pointed out above this combination of representations and abstractions is desired so that a student is fully capable of systematically processing their solution or debugging a problem utilizing a CT approach (Basu, Biswas, & Kinnebrew, 2016).

2.2. General Robotics Courses

Many schools offer after school programs or summer camps using VEX® or LEGO Mindstorms® robotic kits. These kits come with a substantial amount of supporting information and resources including forums, tutorials, and fully executable curriculum sets. Hendricks et al. (2012) and Panadero et al. (2010) report an increase in computational thinking activities and learning outcomes when students use these kits. Other robotics courses offered as summer camps have been successful in increasing student engagement, motivation, teamwork, critical thinking, and problem solving (Darrah, Kuryla, & Bond, 2018; Goldman, Eguchi, & Sklar, 2004; Ansorge & Barker, 2007), all directly related to the application of CT constructs in a STEM domain.

2.3. Robotics in Mathematics

Barreto & Benitti (2012) noted that activities which integrate robotics into a math or science classroom should "*possess a high-level of structure that helps the robot to correctly guide the activities and the students through them*," and that self-directed activities that "*promote personalized comprehension of STEM concepts through experimentation*" showed significant success - and added support for our approach in this domain as design space exploration activity. Our DSML has been highly scaffolded as a means of supporting these robotic integration requirements. In addition, the experimentation requirement is further supported through the display of curricular feedback following the execution of a robot sequence, to be described in Section 6.

Two recent studies were carried out by researchers from NYU that explored the use of a robotic agent to teach geometry to middle school students (Muldner, et. al., 2013; Girotto, et. al., 2016). Their environment consisted of a projector, a LEGO Mindstorms® robot, and two iPods for communication. These studies highlight the effectiveness of a tangible learning environment (TLE) in terms of delivering a much richer learning experience than traditional classroom methods. Moreover, TLEs have found considerable success in fostering creativity (Goldman, Eguchi, & Sklar, 2004), a benefit to our design space exploration approach, while also increasing motivation (Windham, 2007).

2.4. Domain Specific Modeling Language (DSML)

Van Deursen defines a domain specific language as "*a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain*" (2000). Typically, DSMLs are developed to facilitate the work of domain experts in application tasks. But they can also play an important role in helping learners focus on domain concepts when building models and solving problems in the domain. In our work, the DSML developed allows a student to define a set of instructions for a robot to solve middle school mathematics problems that are centered on concepts derived from coordinate geometry and solving path planning problems.

The benefit of developing a DSML is the affordability it creates in curricular implementation and expansion. Students can "*express and develop solutions … at the level of abstraction of the target domain*," "*build programs that are concise and self-documenting*," and "*verify and validate models and results generated from the models*" (Hasan & Biswas, 2017). This provides a highly structured environment that enables the student to experiment with various solutions in a self-directed manner. This structure comes in part by how the model building environment is presented to the student (visualizer), how the model blocks themselves appear (decorator), and how the model is executed on the robot (communication plugin), to be detailed in Section 5.

Jackson and Sztipanovitz (2008) highlight three applications of DSML syntax: model transformations, correct-by-construction, and design space exploration. In the context of an educational setting, students engage with a robotics-based design studio to learn mathematics and CT concepts by performing tasks with their robots. The syntax our DSML most closely supports is the notion of design space exploration. This enhances "*the expressiveness of metamodeling constraints" and the ability "to project behavioral properties on the syntactic level*" (2008). Our robotics DSML supports model building and problem solving with robotics in a way that students can seamlessly learn domain and CT concepts and practices.

As it relates to our DSML development, we aimed to simplify the interactions between the robot and the students, so they may focus on learning the required mathematics and geometry concepts and applying them to planning and problem-solving tasks. An added goal is to provide for easy exploration within the domain, so that the open-ended nature of the learning is retained, and students can learn through the direct application of CT practices such as model construction and algorithm development.

Finally, as an educational product, it is imperative to understand the ramifications this implementation has on teacher curriculum development and productivity in the classroom. In Tennessee, the licensure and examination process does not require any assessment of computer science or CT knowledge (The Praxis Study Companion, 2017). As such, we assume limited CS experience of middle school mathematics teachers. To account for this, our DSML can be tailored at the classroom level to account for the capabilities of the teacher. This flexibility eases the transition from learning the system to learning the instructional material the system delivers.

3. CURRICULUM DEVELOPMENT

Understanding how students conceptualize, acquire, and retain geometric concepts must be understood in sufficient detail before designing a curriculum in conjunction with a TLE. Burger and Shaughnessy (1986) concluded that there are five major stages to student's understanding of geometric concepts: visualization (pure visual reasoning), analysis (based on visualization), abstraction (understanding the properties), deduction (formal reasoning), and rigor (comparing different systems). Students are not typically exposed to deduction or rigor until a high school geometry course.

We focus primarily on visualization, analysis, and abstraction by introducing a new concept with a description, graphic, and how this topic is relevant in a student's everyday life. Then we provide a set of problems in which the student must give the robot the correct information so it can achieve its goal. Geometric properties and definitions are introduced with their respective problems, and students are required to not only demonstrate mastery by generating the correct command sequences, but also with summative assessments at the end of each module. Below is a sample curriculum outline that is well suited for middle school geometry:

1) Coordinate Plane (Axis definitions, Points)
2) Lines (Properties, Line segments, Slope, Midpoints)
3) Shapes (Properties, Squares, Rectangles, Triangles)
4) Path Planning (Shortest path reasoning, Manhattan distances, Straight line distances)

As described in the introduction and requirements, our goal with the development of a robotics DSML was to provide the basis to enable an engaging, applicable curricular unit for a middle school mathematics classroom that connects the computational modeling task to modeling and problem solving in geometry. Our new learning environment promotes knowledge acquisition through a hands-on, visual-feedback approach that is consistent with the design of TLEs (Darrah, Kuryla, & Bond, 2018) and linked to the visualization, analysis, and abstraction stages of geometry understanding described by Burger and Shaughnessy. Our development of a model via WebGME (given the abstraction afforded in the DSML) with the added benefit of watching a real-life robot complete the programmed paths allows for easy applications of CT and geometry constructs and students will be more motivated by the experience.

As it pertains to CT learning gains, our curriculum is most applicable to the assessment of students' knowledge and abilities in implementing algorithms, understanding and addressing efficiency and performance constraints, and debugging. These practices, as defined by Grover and Pea (2013), are utilized in each curricular task designed to target the elements provided in the curriculum outline, above, as students are required to use our scaffolded DSMLs in a sequential order given physical and command constraints of the robot in order to complete each task. We surmise that the repetitive use of these practices to solve geometry problems will enhance students CT abilities for these practices.

4. ENVIRONMENT

With the establishment of our system requirements, the second step in our process was the design and development of our system environment. Our robot sits on a 7ft by 7ft platform that has been sectioned into a 10x10 grid. The robot is equipped with sensors that allow it to track its location on the grid. As such, if it is told to move forward by 3, the robot will travel forward until it has reached the third black line that is perpendicular to the direction the robot is moving. A video camera set-up is centered above the grid as shown in the figure. The video feed generated can be used by the student or a teacher to track the robot as it moves along a path and verify the correctness of the path.

4.1. Robot

When activated, the robot starts a TCP server to communicate with the WebGME plugin and opens a serial port to communicate with the Arduino MCU. It manages these processes on separate threads. The main thread manages the various modes the user can utilize to control the robot, such as manual mode, sequence mode, or GME mode (the mode used in conjunction with this paper). The MCU runs one program that takes input from 3 IR tx/rx modules (line following sensor) and its output controls the motors. It communicates with the SBC as well to provide feedback for received commands and for mode switching.. Figure 2 provides an overview of the modular system architecture.

The robot communicates with WebGME using the cross-platform *socketio* library. The plugin generates a JSON formatted string that is parsed within a minimal Flask web server running on the robot. Upon receipt, the Arduino MCU executes the command sequence and signals to the RCM when it is finished.

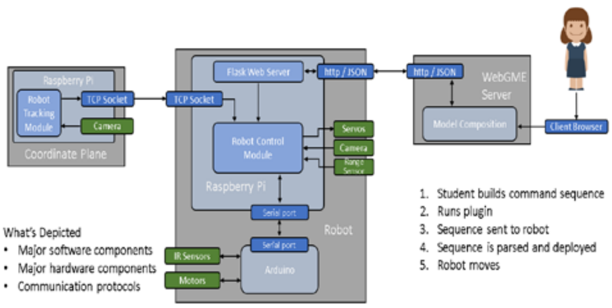

Figure 2. System Architecture

5. META-MODEL

As previously described, the utilization of a DSML provides curricular benefits in that it is constructed at a suitable level of abstraction to allow the learner to focus on what is important, and abstract away other CS details (e.g. syntax concerns). Through the analysis of geometry and CT requirements, our meta-model (Figure 3) was developed based on the implementation of four goals:

1. a scaffolded, curricular driven approach that focuses student actions on the concept(s) being addressed;
2. a simplified integration of robotics and mathematics that makes it easier for the teacher to follow the student work and assess it;
3. scalability in the classroom context; and
4. a systematic, stable connection between the robot environment and modeling environment that is easy to understand.

The students' problem-solving tasks (e.g., building shapes, following paths) are scaffolded, as exemplified through the four available commands. The reduced set of commands allows students to focus on the planning and computational components of their activities. In addition, the organization of the commands and sequences showcases the model's potential scalability and ease-of-use for the teacher.

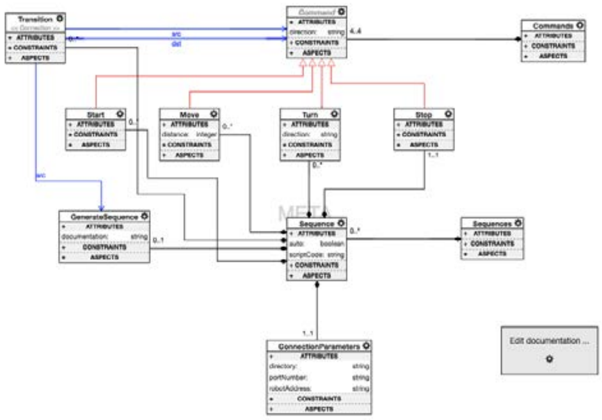

Figure 3. Robotics Meta-Model

5.1. Decorator

The target audience for this activity includes middle school students that may not have any programming experience. As such, the visual component of the environment may play a role in the motivation and buy-in of students, regardless of their capabilities, which is directly linked to positive learning outcomes. A Decorator is a component of the WebGME Design Studio that alters the way a node in the model looks in composition view (the student's view). Figure 5 provides a zoomed-in image of relevant decorator components. Students can select the next command in their sequence via a drop-down menu located on the current node. When a command is selected, the transition between the two nodes is automatically created. In addition, each node contains the command name, attribute value, and an image - not only allowing for multimodal learning acquisition, but also easing the debugging process described in Section 2.2.

Figure 5. Model Decorator

5.2. Plugin

The final component needed to configure our WebGME design studio is the plugin that coordinates the compilation and delivery of the sequence of commands implemented by the student to be executed by the robot. In other words, the JavaScript plugin sends the visually represented sequence of commands to the robot in a machine-readable format. In the making of the plugin, we defined three requirements: Parsing the student defined command sequences into a standard structure, validating the sequence alongside reporting the errors, and finally, sending the commands to the robot.

Upon starting a session, the plugin connects the editor environment with the robot using the parameters defined in the "Connection Parameters" node. This is achieved through a one-to-one socket connection, which remains open until the user ends the session. To make sense of the visual chain of commands the plugin starts by querying the sequence to find the start node. It then records this block and its relevant attributes. Next, the outgoing connection is followed to similarly parse the next blocks until the stop command is reached. This information is then stored in JavaScript Object Notation (JSON) format and sent to the robot by emitting a submission event that the robot is listening for. The robot then parses the sequence and executes the commands as detailed above.

6. Implementation

Following the development and design of the robotics studio and accompanying geometry curriculum, we had three middle school students complete the designed tasks as a means of testing the system and getting feedback on ease-of-use and system benefits or drawbacks. In this section, we present an application of our system in a classroom environment and demonstrate the use of the robotics design studio as a tool to complete a sample path planning module at the middle school level.

6.1. Sample Problem Set

A subset of the curriculum described in Section 3 includes three general problems:

1) Identifying the axes and positive or negative values
2) Plotting points given (x,y) and deriving (x,y) from a set of points
3) Path planning with multiple points, calculating the shortest Manhattan distance

Figure 6 illustrates the visual interface that provides the instructions for each task along with the overhead webcam feed in conjunction with the WebGME design studio. In this assignment, students are tasked with finding the most efficient path the robot can take ensuring stops at the police station, the fire station, and the courthouse prior to ending its trip at the post office. Typically, this type of assignment at the introductory level is distributed as on paper, limiting the multi-modal approach to learning that may benefit certain students.

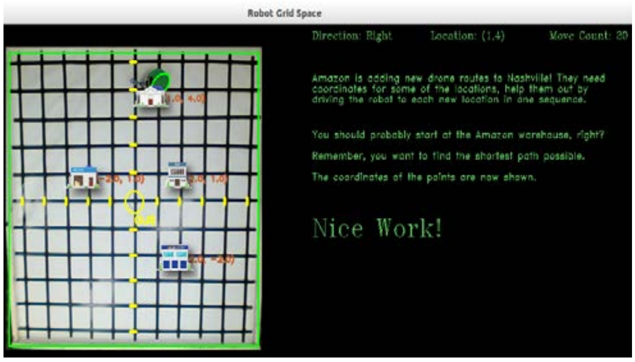

Figure 6. Virtual Interface for Example Path Planning Problem

The direction the robot is facing, its current location, and number of spaces moved are displayed at the top of the information section which helps the student during the solution construction process. The problem is given below that, along with various hints that are given at predetermined times.

In the scenario shown in Figure 6, the student first identified the coordinates of all locations the robot must visit. When all points are correctly located, their coordinates are shown on the video feed. From the image provided, it can be seen that the student then completed a shortest path problem in which they generated the correct command sequence for the robot to visit all locations, starting at the Amazon warehouse (2, -2). The automatic feedback response of "Nice Work!" is shown – demonstrating the successful completion of the task

In Figure 7, the solution to the above problem is shown. Upon closer inspection, the distance values can be seen as well. Sequences can become significantly long, making the debugging process difficult should an error occur in the robot's path. The availability of the command name and attribute value as text on the node as well as images of blocks allow for an easier analysis of the complete path during the debugging process.

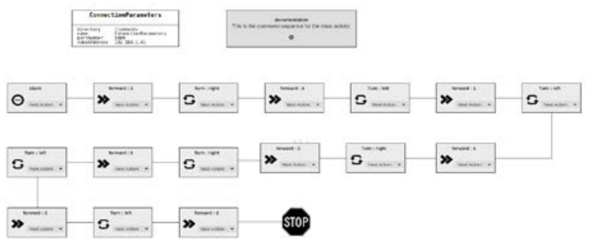

Figure 7. Student Solution to the Path Planning Problem

6.2. Case Study: CT Gains

For our usability study, students were asked to complete a pre- and post- challenge. The challenge contained two parts: the first included a debugging task in which they were asked to analyze a given robot sequence and improve the efficiency of the sequence while also ensuring the end location was correct. This challenge component was designed to assess student abilities in the CT constructs of flow of control and debugging. The second task involved the development of a sequence that would allow the robot to draw a given shape with the minimum commands possible in the grid space depicted in Figure 1, thereby assessing student understanding of efficiency and performance constraints as well as another application of flow of control. This pre- and post- nature of the challenge was implemented to identify potential improvements in applying these CT constructs.

S1 is a 13-year-old middle school male student and S2 is a 14-year-old middle school female student. Both students identified as having little to no experience with the listed geometry concepts and practices and both identified as having some previous programming experience using block-based programming languages. For the purpose of this case study, we will focus on student work in part 1 of the challenge.

In the pre-challenge, S1 and S2 failed to debug the given path in Part 1 in a manner that provided the fastest path for the robot to complete the task. In addition, both S1 and S2's robot sequences could not make the robot arrive at the correct location, indicating that both students struggled to debug the entire algorithm. However, S1 and S2 were able to identify two of the five identified errors indicating that they had a preliminary understanding of flow of control.

Following the geometry assignments, S1 and S2 completed the robotics post-challenge. This time, S1 was able to identify three of the five identified errors and the final sequence allowed the robot to finish at the desired location. It should be noted that the student drew a path on the given image of the grid that accounted for the two missing errors in the algorithm, but those errors were not identified in the algorithm. As S1 was able to identify the most efficient path in the image, we believe it may be necessary for us to assess how we described the challenge in order to be as clear as possible on how each student should define his or her response.

S2's approach to Part 1 of the post-challenge changed significantly from the pre-challenge. In Part 1 of the post-challenge, S2 drew her robot's shortest path sequence on the grid provided, with dots along the grid indicating that she was counting various path options (an action she commonly did with her finger via the virtual interface during the geometry assignments). While her new path followed the expert model path between a few specified target points, a few sub-paths were significantly different than the expert model path. However, her final path was shorter than the given problem to debug and one away from the shortest path possible. Given her search-based, debugging approach in the post-challenge, it can be seen that her utilization of CT constructs improved.

6.3. Case Study: Geometry Gains

Our final student, S3, reported significant experience with block-based programming environments like Scratch and Netsblox. S3 achieved a perfect score on the CT related questions of the pre-challenge. A key point here should be made - S3 is younger than both S1 and S2, who report no experience with DSMLs, and outperformed them both on the pre-challenge, supporting our hypothesis that DSMLs are linked to the utilization of CT strategies when solving problems. During the geometry tasks, S3 initially struggled with the coordinate plane unit, including the identification of quadrants and moving the robot to desired x,y points on the plane. However, this student made use of the system feedback given. After repeating similar tasks, the time spent solving coordinate plane tasks decreased. Based on these observations, it can be seen that while learning gains in CT could not be measured due to the perfect pre-challenge score; abilities in geometry improved.

7. Results and Future Implications

This paper details the theoretical and systematic design and development process of a robotics DSML for use in a middle school mathematics classroom. Through an analysis of curricular and software requirements, our group implemented a robotics design studio using WebGME that allows for an applicable and scalable robotics activity to support CT and STEM learning. In addition, our usability studies indicate potential CT learning gains acquired through the completion of the geometry curriculum in our environment. The potential benefits of integrating robotics into other STEM classrooms has not been actualized to the extent that it was theorized by renowned educational theorist Seymour Papert (1993). The application of this highly scaffolded DSML in a middle school classroom may allow for a fruitful analysis on the level or extent of programming needed to not only advance CT learning and understanding, but also ensure the successful delivery of relevant STEM content.

8. Acknowledgements

We would like to thank Patrik Meijer, Tamás Kecskés, and other collaborators from Vanderbilt University for their numerous contributions. This research is supported by NSF grant # IIS 1735909.

9. REFERENCES

Anderson, J.R., Boyl, C.F., Corbett, A.T., Lewis, M.W. (1990). Cognitive Modeling and Intelligent Tutoring. Artificial Intelligence - Special issue, 42-1.

Ansorge, J., Barker, B. (2007). Robotics as a Means to Increase Achievement Scores in an Informal Learning Environment. Journal of Research on Technology in Education, 39-3.

Basu, S., Biswas, G., Kinnebrew, J.S. (2017). Learner modeling for adaptive scaffolding in a Computational Thinking-based science learning environment. User Modeling and User-Adapted Interaction, 27(1), 5-53.

Basu, S., Biswas, G., Kinnebrew, J. (2016). Using multiple representations to simultaneously learn computational thinking and middle school science. Proceedings of the 30th AAAI Conference on Artificial Intelligence.

Benitti, F. & Barreto, V. (2012). Exploring the educational potential of robotics in schools: A systematic review. Computers & Education, 58(3), 978-988.

Burger, William F., Shaughnessy, J. Michael: Characterizing the van Hiele levels of development in geometry. Journal for research in mathematics education, p. 31-48. (1986)

Darrah, T., Kuryla, E., & Bond, A. (2018). Improving STEM Education with an Open-Source Robotics Learning Environment. Proceedings of the Hawaii International Conference on Education.

Girotto, V., Lozano, C., Muldner, K., Burleson, W., Walker, E. (2016). Lessons Learned from In-School Use of rTag: A Robo-Tangible Learning Environment. Proceedings of the ACM Conference on Human Factors in Computing Systems.

Goldman, R., Eguchi, A., Sklar, E. Using Educational Robotics to Engage Inner-City Students with Technology. (2004). Proceedings of the 6th International Conference on Learning Sciences, 214-221.

Grover, S. & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. Educational Researcher, 42(1), 38-43.

Hasan, A. & Biswas, G. (2017). Domain Specific Modeling Language Design to support Synergistic Learning of STEM and Computational Thinking. In Proceedings of the International Conference on Computational Thinking Education.

Hendricks, C., Alemdar, M., Olgetree, T. (2012). The Impact of Participation in Vex(R) Robotics Competition on Middle and High School Students' Interest in Pursuing STEM Studies and STEM-Related Careers. American Society for Engineering Education.

Hutchins, N, Zhang, N, & Biswas, G (2017). The Role Gender Differences in Computational Thinking Confidence Levels Plays in STEM Applications. In Proceedings of the International Conference on Computational Thinking Education.

Jackson, E. & Sztipanovits, J. (2008). Formalizing the Structural Semantics of Domain-Specific Modeling Languages. Software & Systems Modeling, 8(4), 451–478.

Jensen, J. C., Chang, D. H. Lee, E. A. (2011). A Model-Based Design Methodology for Cyber-Physical Systems. Proceedings of the IEEE Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems.

Jona, K., Wilensky, U., Trouille, L., Horn, M. S., Orton, K., Weintrop, D., & Beheshti, E. (2014). Embedding computational thinking in science, technology, engineering, and math (CT-STEM). In future directions in computer science education summit meeting, Orlando, FL.

Muldner, K., Lozano, C., Girotto, V., Burleson, W., Walker, E. (2013). Designing a Tangible Learning Environment with a Teachable Agent. Artificial Intelligence in Education.

National Research Council. (2012). A framework for K-12 science education: Practices, crosscutting concepts, and core ideas. National Academies Press.

Panadero, C., Villena-Roman, J., Delgado-Kloos, C. (2010). Impact of Learning Experiences Using LEGO Mindstorms(R) in Engineering Courses. Proceedings of the IEEE Global Engineering Education Conference.

Papert, S. (1993). Mindstorms: Children, computers, and powerful ideas (2nd ed.). New York, NY: Basic Books.

Royal Society. (2012). Shut down or restart: The way forward for computing in UK schools. Retrieved February 4, 2017, from https://royalsociety.org/topics-policy/projects/computing-in-schools/report/

Sengupta, P., Kinnebrew, J.S., Basu, S., Biswas, G., & Clar, 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.

The Praxis Study Companion - Mathematics: Content Knowledge. ETS, 2017.

Van Deursen, A., Klint, P., & Visser, J. (2000). Domain-specific languages: An annotated bibliography. Sigplan Notices, 35(6), 26-36.

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*, 1–21.

Windham, C.: Why Today's Students Value Authentic Learning. (2007). Educause Learning Initiative - Advancing Learning Through IT Innovation.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-36.