Analyzing debugging processes during collaborative, computational modeling in science

Nicole M. Hutchins, Vanderbilt University, nicole.m.hutchins@vanderbilt.edu
Caitlin Snyder, Vanderbilt University, caitlin.r.snyder@vanderbilt.edu
Mona Emara, Damanhour University, mona.emara@damanhour.edu.eg
Shuchi Grover, Looking Glass Ventures, shuchig@cs.stanford.edu
Gautam Biswas, Vanderbilt University, gautam.biswas@vanderbilt.edu

Abstract: This paper develops a systematic approach to identifying and analyzing high school students' debugging strategies when they work together to construct computational models of scientific processes in a block-based programming environment. We combine Markov models derived from students' activity logs with epistemic network analysis of their collaborative discourse to interpret and analyze their model building and debugging processes. We present a contrasting case study that illustrates the differences in debugging strategies between two groups of students and its impact on their model-building effectiveness.

Introduction

Research has demonstrated debugging's pivotal role in learning computational thinking (CT; Rich et al., 2019) and learning by computational modeling in science domains (Sengupta et al., 2018), thus warranting further research to enhance our understanding of debugging processes that support this learning. Analyzing students' performance purely on the basis of the artifacts (model, program) they produce may not provide sufficient information of the processes they apply for learning computational concepts and practices, and, therefore, the difficulties they face in their model-building tasks (Grover et al., 2017). In addition, analyzing activity data from log files may not provide sufficient evidence on the processes students employ to integrate domain-specific knowledge with CT concepts and practices to build and debug models. To overcome this problem, we develop a multimodal learning analytics (MMLA) approach combining quantitative activity data and collaborative discourse analysis to study students' debugging processes during collaborative, computational modeling activities in a high school physics class. We focus on the following research questions: (RQ1) *How can we identify and characterize students' model-building and problem-solving processes from the log data collected as they worked on modeling science phenomena?* and (RQ2) *What additional information can we derive from students' collaborative discourse on the processes they employ in integrating science and CT concepts and practices as they build and debug their computational science models in a block-based programming environment (BBPE)?*

MMLA framework to evaluate collaborative debugging

A review of recent national standards in science, technology, engineering, and math (STEM) and computer science demonstrates that common problem-solving practices targeted across the two domains include the creation of artifacts and using CT (NGSS, 2013; Collegeboard, 2017). Students' application of these practices offers a unique context to examine their learning processes as they build computational science models. We hypothesize that analyzing instances of shared practices will provide an informative context for examining mutually beneficial instances of students' synergistic STEM and CT learning (Snyder et al., 2019). We focus on two factors that are indicative of synergistic learning: (1) shared practices between STEM and CT and (2) combining concepts through domain-specific block structures provided in our environment (e.g., contextualizing a conditional block when considering the stopping condition for a truck to stop at a stop sign) (Hutchins et al., 2020).

Our analysis centers on the examination of students' model-building and debugging strategies in C2STEM, a web-based, BBPE (Hutchins et al., 2020). In this study, 14 high school students worked in groups of 2 or 3 to build multiple kinematic models of 1-D and 2-D motion of objects. The curriculum included instructional tasks, designed to help students develop specific physics and CT concepts, followed by model-building tasks where students modeled the motion of a truck in three phases: (1) a speed up phase; (2) a cruise phase with constant velocity motion; and (3) a slow-down phase where the truck comes to a stop at a pre-designated STOP sign. Students were encouraged to work together to understand the problem, and build a kinematics model that produced the right behaviors.

To answer RQ1, we applied Markov Chain analysis (Craig & Sendi, 2002) to logged activity data for each student group and evaluated the occurrence of high probability sequences of actions and represented their temporal occurrence using CORDTRA graphs. Student actions were recorded in log files with timestamps, and analyzed sequentially along with the context in which they were being applied to derive action patterns that we then mapped into model-building and debugging tasks. To increase interpretability, we developed a hierarchical task-oriented structure (Emara et al., 2021) to map student actions to a level of abstraction so that the derived action patterns could be interpreted as code construction and code assessment activities (cf. Rich et al., 2019). Code assessment actions were labeled as either data analysis (DATA; this involved opening the graph tool) or visual feedback (PLAY; i.e., running the simulation). Code construction actions were labeled as either building the model (BUILD; i.e., constructing new elements of the program code) or adjusting the existing model (ADJUST; i.e., debugging code after identifying an issue).

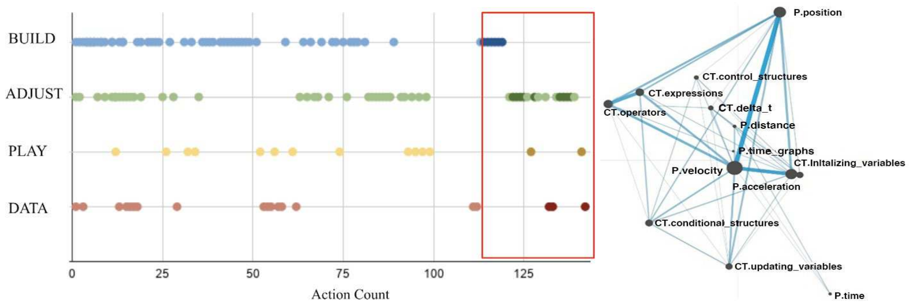
To answer RQ2 and track students' ability to understand and combine CT and STEM concepts and practices, we developed a MMLA approach that combined our log-file based behavioral pattern analysis with Epistemic Network Analysis (ENA; Shaffer et al., 2009) of students' discourse. As a result, our MMLA analysis helped us better interpret temporal patterns derived from coded discourse data on STEM and CT topics. Our coding scheme included physics concepts: P.position, P.velocity, P.acceleration, P.time, P.distance, P,displacement and P.time_graphs. Similarly, the coding scheme included the following CT concepts: CT.delta_t, CT.block_ordering, CT.control_structure, CT.initalizing_variables, CT.updating_variables, CT.operators_expressions, CT.conditional_structures, CT.data_collection and CT.data_visualization. Two researchers transcribed the audio for the two case study groups selected and coded 40 utterances (approximately 15% of all segments) using this coding scheme, resulting in good IRR agreement (k = 0.76). One researcher coded the remaining segments. We present a mixed-method case study approach to illustrate two contrasting cases, where students adopt different approaches resulting in varying degrees of success in their model building tasks. Groups were scored utilizing our integrated STEM+CT computational modeling rubric (Emara et al., 2021) and the two groups selected represent high performing and low performing groups based on a median split of the overall, final model scores. We hypothesize that students who are successful in integrating STEM and CT concepts also show greater learning in both domains.

Results and discussion

Results of the Markov Chain (MC) analysis revealed students used key modeling and debugging strategies, including: (1) **Depth-First** (DF) for multiple, simultaneous code construction actions (e.g., a sequence of BUILDs) without intervening code assessment actions (e.g., PLAY actions), (2) **Tinkering** (TIN), or trying different parameters or block changes in a trial and error fashion to gain some understanding of how to get to a solution (e.g., sequences of "ADJUST→PLAY" (0.20) and "PLAY→ADJUST (0.39) actions), (3) **Multi-Visual Feedback** (MV), represented by sequence of PLAY activities with high probability (0.19), and (4) **Simulation-based Assessment** (EVAL), or evaluating the model using the data tools, e.g., PLAY→DATA (0.29) and DATA→PLAY (0.19). We believe this may represent a build and test strategy. A more refined version of build and test, the modeling-in-parts strategy, has also been shown to support successful model creation (c.f., Grover et al., 2017). This strategy involves working on the model in small segments (i.e., BUILD↔ADJUST) combined with a sequence of execution actions (PLAY↔DATA) to understand the behaviors generated by that segment of the model, and debugging the model, as needed. We will examine identified processes in our case study.

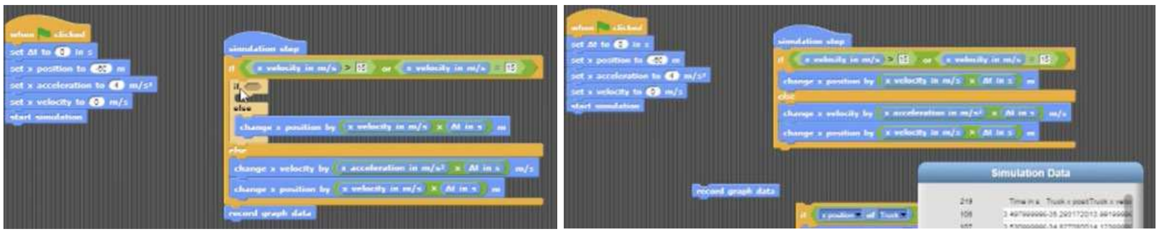
We apply our MMLA framework to examine the reasoning processes of two groups of students that had difficulties in constructing the conditional expression for the third phase of the truck's motion. Group 1 is characterized by their DF approach to model construction and debugging. Figure 1(a) shows that before action #50, most of the model building happened in a DF manner. The DF approach resulted in complex conditionals (Figure 2(a)), which increased the complexity of their debugging tasks. Rather than using feedback from the tools provided to analyze their model behaviors (i.e., PLAY or DATA actions), they used their physics knowledge to analyze the model (via pen-and-paper calculations), and a DF approach (e.g., the darkened blue and green circles in the red box of Figure 1(a)) in an attempt to make their code correspond to the correct physics solution they had derived on paper. These results match the literature on debugging that has shown novices tend to use depth-first construction and debugging strategies because they lack the insight for breaking down a complex task into its sub-parts (Grover et al., 2017). The group's ENA graph (Figure 1(b)) indicates a separation of physics-focused and CT-focused discourse with the top three connections between concepts in the ENA graph include "position-velocity" (edge weight = 0.78), "velocity-acceleration" (edge weight = 0.70), and "operators-expressions" (edge weight = 0.62). The inability to think of the physics and CT concepts as one integrated unit, may indicate difficulties in translating their pen-and-paper physics calculations into computational form (Basu et al., 2017).

The group was unable to build a correct model. Instead of working on the conditional expressions, the group used the data tools and analyzed parts of the model (Figure 2(b)) to calculate the time required from start to cruise (the speed up phase), and then used that information to calculate on paper the distance at which to slow down. This further confirmed their strength in domain knowledge, but weakness in applying CT-related concepts.



(a) (b)

Figure 1. Group 1 CORDTRA graph (a) and ENA graph (b).



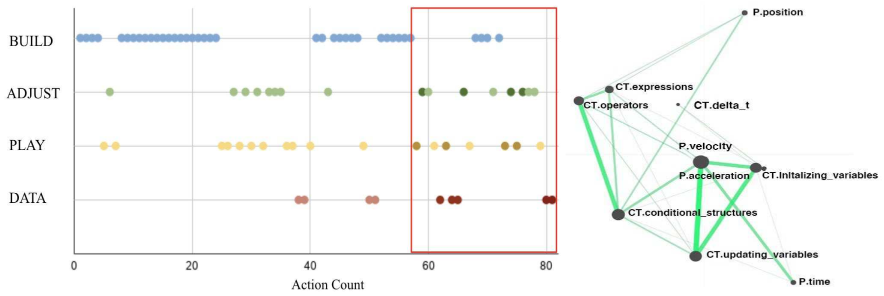
```
when [flag] clicked
set Δt to [illegible] in s
set x position to [illegible] m
set x acceleration to 4 m/s²
set x velocity to 0 m/s
start simulation

simulation step
if < x velocity in m/s > 15 > or < x velocity in m/s = 15 >
    if
    else
        change x position by ( x velocity in m/s × Δt in s ) m
else
    change x velocity by ( x acceleration in m/s² × Δt in s ) m/s
    change x position by ( x velocity in m/s × Δt in s ) m
record graph data
```

```
when [flag] clicked
set Δt to [illegible] in s
set x position to [illegible] m
set x acceleration to 4 m/s²
set x velocity to 0 m/s
start simulation

simulation step
if < x velocity in m/s > 15 > or < x velocity in m/s = 15 >
    change x position by ( x velocity in m/s × Δt in s ) m
else
    change x velocity by ( x acceleration in m/s² × Δt in s ) m/s
    change x position by ( x velocity in m/s × Δt in s ) m

record graph data

if < x position of Truck >
```

(a) (b)

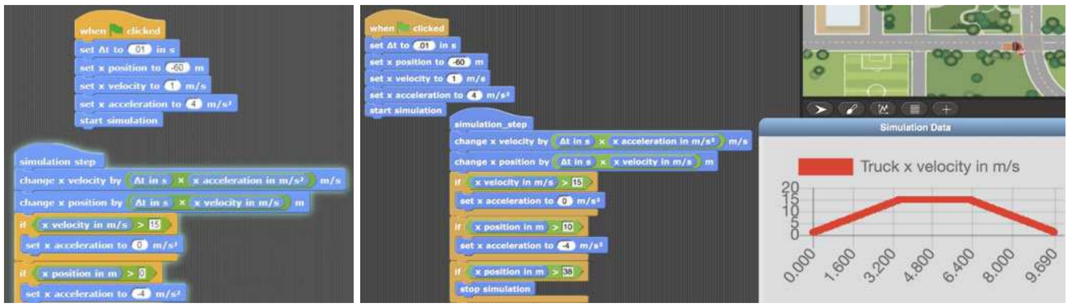
Figure 2. Group 1 code snippet prior to debugging (a) and after debugging (b).

Group 2 used a modeling-in-parts strategy to construct their model. Their debugging processes started around action 60 (indicated by the red box in Figure 3(a)), and it included multiple PLAY and DATA actions, indicating the use of TIN and EVAL strategies identified previously. Simultaneously, their ENA graph (Figure 3(b)) indicates increased integration of STEM and CT discourse elements compared to Group 1. The top three discourse connections were "updating variables - velocity" (edge weight = 0.87), "updating variables - acceleration" (edge weight = 0.73), and "operators - conditional structures" (edge weight = 0.73).



(a) (b)

Figure 3. Group 2 CORDTRA graph (a) and ENA graph (b).



```
when [flag] clicked
set Δt to .01 in s
set x position to -60 m
set x velocity to 1 m/s
set x acceleration to 4 m/s²
start simulation

simulation step
change x velocity by ( Δt in s × x acceleration in m/s² ) m/s
change x position by ( Δt in s × x velocity in m/s ) m
if < x velocity in m/s > 15 >
    set x acceleration to 0 m/s²
if < x position in m > 0 >
    set x acceleration to -4 m/s²
```

```
when [flag] clicked
set Δt to .01 in s
set x position to -60 m
set x velocity to 1 m/s
set x acceleration to 4 m/s²
start simulation

simulation_step
change x velocity by ( Δt in s × x acceleration in m/s² ) m/s
change x position by ( Δt in s × x velocity in m/s ) m
if < x velocity in m/s > 15 >
    set x acceleration to 0 m/s²
if < x position in m > 10 >
    set x acceleration to -4 m/s²
if < x position in m > 38 >
    stop simulation
```

(a) (b)

Figure 4. Group 2 code snippet prior to debugging (a) and after debugging (b).

Unlike Group 1, Group 2 elected to use the visualization tool to adjust the segment of code they were working on. For instance, during their debugging process a student noted that they were not sure if the conditions they created for switching phases would overlap, but they should run the newly created condition with 0 "for now," just to check. Once they determined how that part of their code worked, they used the data tools to determine the approximate location to begin the slowing down phase, recalling that the slowing down process was a complement of the speeding up process that they had implemented. Once they got that condition working, using the data tools to estimate the x-position in which to begin slowing down, they proceeded to implement the final stopping condition, and again used the TIN and EVAL strategies to complete their model (Figure 4(b)).

Conclusions

Our MMLA analyses provides initial evidence that: (1) the analysis of students' modeling and debugging processes by combining activity logs and ENA for discourse analysis provides a deeper understanding of students' reasoning with STEM and CT concepts and practices, and (2) students use multiple approaches and different strategies to develop and debug solutions, and this provides them the opportunities for developing higher order thinking and problem-solving skills. However, the difficulties students face in working on these complex problems also suggest that it may be helpful to provide scaffolding to students when they cannot progress in their problem-solving tasks (Basu et al., 2017). These findings could be used to provide feedback or support for students to make it easier to decompose a complex task into sub-parts and to understand how to employ visualization and data to support debugging. Overall, this approach supports learning-by-modeling approaches for synergistic STEM and CT learning. Finally, these findings can be extended to CS and introductory programming classrooms as well, to guide teaching and learning of programming and debugging.

References

Basu, S., Biswas, G. & Kinnebrew, J.S. (2017). Learner modeling for adaptive scaffolding in a Computational Thinking-based science learning environment. *User Model User-Adap Inter*, *27*, 5–53.

CollegeBoard. Advanced placement computer science principles course guide. 2017.

Craig, B.A., & Sendi, P.P. (2002). Estimation of the transition matrix of a discrete-time markov chain. *Health economics*, *11*(1), 33–42.

Emara, M., Hutchins, N.M., Grover, S., Snyder, C., & Biswas, G. (2021, in press). Examining Students' Regulation of Collaborative, Computational, Problem-Solving Processes in Open-Ended Learning Environments. *Journal of Learning Analytics*. SOLAR.

Grover, S., Basu, S., Bienkowski, M., Eagle, M., Diana, N., & Stamper, J. (2017) A framework for using hypothesis-driven approaches to support data-driven learning analytics in measuring computational thinking in block-based programming environments. *ACM Trans. Comput. Educ*., *17*(3).

Hutchins, N.M., Biswas, G., Maróti, M., Lédeczi, Á., Grover, S., Wolf, R., … & McElhaney, K. (2020a). C2STEM: a System for Synergistic Learning of Physics and Computational Thinking. *Journal of Science Education and Technology*, 29(1), 83-100.

NGSS Lead States. (2013). Next Generation Science Standards: For states, by states. Washington, DC: National Academies Press.

Rich, K. M., Strickland, C., Binkowski, T. A., & Franklin, D. A K-8 debugging learning trajectory derived from research literature. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 745–751, New York, NY, USA. Association for Computing Machinery.

Sengupta, P., Dickes, A., & Farris, A. (2018). Toward a phenomenology of computational thinking in stem education. In *Computational thinking in the STEM disciplines*, 49–72. Springer.

Shaffer, D. W., Hatfield, D., Svarovsky, G., Nash, P., Nulty, A., Bagley, E., Frank, K., Rupp, A., & Mislevy, R. (2009). Epistemic network analysis: A prototype for 21st century assessment of learning. *International Journal of Learning and Media*, *1*(2), 33–53

Snyder, C., Hutchins, N., Biswas, G., Emara, M., Grover, S., Conlin, L. (2019). Analyzing Students' Synergistic Learning Processes in Physics and CT by Collaborative Discourse Analysis. *Proceedings of 13th International Conference on Computer Supported Collaborative Learning 2019* (360-367), Lyon, France. International Society of the Learning Sciences.

Acknowledgments

This research is sponsored by the National Science Foundation Award DRL-1640199. We also appreciate the contributions of our C2STEM research team members and the suggestions of the anonymous reviewers.