Examining Students' Debugging and Regulation Processes During Collaborative Computational Modeling in Science

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

Abstract: In this mixed-methods case study research, we examined student discourse and actions through the dual lenses of cognitive and regulation processes to analyze problem-solving processes in the context of computational model building in a high school physics classroom. We conducted descriptive statistical analyses of debugging processes during model building in a block-based programming environment. We also qualitatively examined debugging episodes by analyzing video capture of three groups modeling 2D motion of a boat crossing a flowing river. Our results demonstrate differences between groups in terms of their debugging behaviors and regulation of problem-solving processes that potentially impact the groups' learning outcomes. Our analyses demonstrate the promise of our approach and lay the foundation for guiding future automated approaches to support debugging during computational modeling.

Introduction

Building computational models of science phenomena engages students in construction, inquiry, and interpretation tasks that involve computational thinking (CT) practices like use of modeling abstractions, decomposing complex model-building tasks, building and analyzing models, and debugging these models (Sengupta, et al., 2013). A large portion of students' model-building effort is spent testing, debugging, and refining solutions (Liu, et al., 2017). Students often experience difficulties to adequately debug their models and translate their STEM knowledge into computational model representations (e.g., Hutchins, et al., 2019; Sengupta, et al., 2013). Prior research in the learning sciences has examined cognitive processes of debugging (such as problem identification, bug location and bug correction) to understand how students detect errors and find defects in their computational solutions (e.g. Klahr & Carver, 1988). Recently, the concept of regulation during problem solving—the ability to understand, monitor and productively adapt one's/or other's learning and problem-solving process—has been gaining much attention for its potential value and applicability in STEM curricular frameworks and situations (NRC, 2012; Winne & Hadwin, 1998; Hadwin, Järvelä, & Miller, 2011). However, research on CT has not focused on learner regulation during problem-solving processes such as debugging.

We believe that examining learner regulation during problem solving bears the potential to provide insights into the crucial learning and application of debugging. In this paper, we integrate cognitive processes of debugging with principles of learner regulation into a single conceptual framework to examine debugging processes with greater nuance. This approach has implications on potential methods for supporting and assessing CT. By analyzing both cognitive and regulatory processes during debugging of computational models, we can investigate tools that can scaffold debugging processes of individuals or groups of learners. Given the growing consensus on improving students' CT and STEM learning (Grover, et al., 2019; Peters-Burton, et al., 2018), we need a conceptual framework to dive deeper into the debugging processes students' employ when testing and refining the program to generate a correct computational model.

This paper presents our analysis of debugging episodes of three triads as they work in a high school Kinematics unit in 'C2STEM'—a web-based, computational modeling environment that we designed to support the programming of scientific models for integrated learning of physics and CT for high school students (Hutchins, et al., 2019). We adopted a multi-case study approach to contrast the debugging processes during model building used by the three triads and their learning outcomes in physics and CT using the groups' models scores. The case studies and analyses demonstrate that groups that exhibit more regulation during debugging show greater awareness and better understanding of the computational modeling tasks.

Background

Debugging and Collaborative Problem Solving (CPS)

Computational Thinking is essentially a problem-solving methodology (Grover & Pea, 2018; Wing, 2008) and debugging has been widely considered an essential part of CT and coding. Debugging, defined as the systematic process of identifying why code malfunctions and fixing the cause, is central to programming and computer science, and part of problem solving (McCauley, et al., 2008; Yen, Wu & Lin, 2012) essential to STEM and other learning. In CPS, successful debugging requires several interdependent skills, including encouraging each other's participation, contributing ideas, summarizing each other's contributions, providing constructive feedback on ideas, and monitoring group progress (Goos, Galbraith, & Renshaw, 2002). A particularly critical set of skills involves the use of explanations and justifications to convey ideas to other group members (Chi, et al., 2018). These skills support interactive co-construction of knowledge to make informed problem-solving moves (Emara, et al., 2018; Chi, et al., 2018). Collaboration during scientific model building or while working with science simulations has been studied through examining collaborative conversational interaction to understand conceptual change in dyads working with a kinematics simulation (Roschelle & Teasley, 1995) and co-creation of science models in a web-based science inquiry environment (Gobert, et al., 2007). However, we do not as yet have a comprehensive understanding of the crucial CT skill of *debugging during* ***collaborative computational modeling involving programming***.

A framework for studying regulation of debugging

Lacking a current theoretical framework of self-regulation of learning (SRL) in computing in general, and debugging in particular, we derive our framework from key self-regulation principles which are common across prior work. Across theoretical frameworks, SRL is typically conceptualized as a cyclical problem-solving process whereby learners monitor and control their behaviors, thoughts, and emotions for the demands of the moment, and monitor progress toward goals (Pintrich, 2000). In the cognitive debugging model (CDM), problem solving involves five cognitive phases: problem identification, problem representation, error location and error correction, and program evaluation (from Klahr & Carver, 1988). In the regulation of debugging model (RDM), we propose that regulation helps plan and evaluate progress toward building the model that solves some computational problem. Our hypothesis is that the more a participant regulates their debugging activities, the more successful they will be at solving the problem. This hypothesis demands a more granular view of what "debugging" means in computational modeling, and how self-regulation is related to debugging.

A few studies have attempted to assess students' cognitive and regulation processes while debugging by analyzing students discourse combined with trace methodologies (Liu, et al., 2017; Loksa & Ko, 2016) or a multiple-choice test where learners interacted with a computer agent (Lin, et al., 2015). However, there is lack of clarity in how students' discussions interplay with regulation of their CPS moves during debugging episodes. Recently, Peters-Burton, et al. (2018) argued that self-regulated learning related to CT is viewed as a goal-directed process whereby a learner is required to understand a problem, examine relevant data to inform a solution, develop a solution, and evaluate the solution. *To our knowledge no prior work within learning sciences literature examines talk during computational model building through a cognition and regulation lens to investigate collaborative debugging.* We believe the joint use of CDM and RDM to understand collaborative computational model building is a unique methodological contribution to the learning sciences and CSCL communities that helps further our understanding of interdisciplinarity and has implications for learning and design.

Methods

The research conducted and presented here was guided by the following research questions: (1) *How do groups' debugging behaviors relate to their summative learning gains in STEM and CT?* (2) *Are there any differences among groups' debugging behaviors?* and (3) *Are there any differences among groups' cognitive and regulation processes of debugging when building computational models?*

Participants and curricular context

Our month-long curricular intervention was designed and developed in C2STEM—a block-based computational modeling environment with pre-defined custom 'domain-specific modeling' blocks (e.g. blocks for setting and updating *position*, *velocity*, *acceleration*, *heading*, and such) that help learners focus on physics concepts (Hutchins, et al., 2019). Eighteen 9th grade students (7 female and 11 male; mean age 15 years) in a high school physics classroom in Southern U.S. were assigned to six groups of triads by the teacher. The groups worked on four C2STEM Kinematics modules as a part of a 10-hour curricular intervention (described in detail in Hutchins, et al., 2019) that covered 1-D motion, 2-D motion with constant velocity, and 2-D motion with gravitational forces. The curriculum for our study involved student collaboration and included instructional tasks, designed to focus student modeling on the specific physics constructs, followed by more advanced challenge tasks. The goal of the more advanced challenge tasks (as opposed to our previous curriculum which included a more scaffolded, 4-task approach) was to promote student discourse as groups developed their modeling solutions. Student groups were instructed to work together to build models, and to switch between "driver" and "navigator" roles (Williams, et al., 2002) during the instructional and challenge tasks.

Data collection and analysis

Triads worked at a single computer. Talk and behavior (for example, attention to the other student) was recorded using OBS™ screen-capture software that recorded C2STEM actions, video and audio. Because the goal of this study was on understanding debugging processes, we focused on data collected while groups of triads (‘S1’, 'S2' & ‘S3’) engaged in a challenge task requiring collaborative model-building related to 2-D motion with constant velocity. This task required students to program (model) a boat to cross a flowing river, making stops at two islands at differing *x* and *y* coordinates along the way. We selected this assignment because it included larger amounts of debugging than other tasks. To answer RQ1, the primary data source was a 268-minutes video recording of the six groups. To answer RQ2 and RQ3, we purposefully selected three case study groups based on a maximum variation criterion which is common in qualitative studies (Creswell & Creswell, 2017). These cases represented groups with different performance outcomes in the modeling task.

Scoring model-building

For scoring the model-building tasks, we used the rubric outlined in Table 1 to define and assess key learning objectives in physics and CT from the models that students constructed (updated from Basu, et al. 2018). The rubric is divided into use of physics and CT constructs in order to evaluate proficiency in each domain separately. The total score of CT and physics models is 10 and 8, respectively.

Table 1: Rubric for evaluating students’ models

PHYSICS COMPONENT:	**points**
Program expresses correct relations among acceleration, velocity, position and time, and correct units for each (x and y motion)	2
Program reflects the effect of the river’s velocity on the boat’s velocity.	3
Program accurately models the behavior of the object needed to answer the task submission question (heading of boat calculated per instance).	3
CT COMPONENT:	
Program makes the distinction between actions that need to happen once during initialization and actions that need to be repeated in the simulation step	1
Program correctly determines which action always happens and which happens under certain conditions, per instance; Stopping condition implemented	4
Program updates the variable corresponding to the sprite’s motion: a. under the correct conditions, and b. in the correct fashion (at every simulation step)	4
All code in the program is reachable and can be executed; No duplicate code	1

Identifying episodes of debugging and activities

For the first stage of the analysis, all the screen-capture video-audio recordings were divided into segments targeting debugging episodes during the model-building process. At this stage, the analysis separated debugging from other types of modeling (e.g., initialization or off-topic activity) that occur during collaborative problem solving. Inter-rater reliability for debugging episodes was established by calculating Cohen’s kappa and resulted in very good agreement (k = 0.87) . For the second stage of the qualitative video analysis guided by Derry et al. (2010), InqScribe software was used to identify, annotate, and analyze time-stamped segments reflecting groups' activities during debugging episodes. The focus of this stage of the analysis was to make note of when students clicked the green flag in the Snap! environment to ‘run’ or test (play) the program (simulation) and compare groups according to the total time spent in debugging, number of debugging actions, and mean of debugging duration from the one simulation ‘Run’ to the next ‘Run’.

Identifying cognitive and regulation processes in debugging

For this stage of analysis, discourse threads were analyzed to understand how the students collectively identified and attempted to solve the debugging problem. Guided by the joint use of CDM and RDM (based on indicators in Klahr & Carver, 1988; Winne & Hadwin, 1998), discourse threads were analyzed according to the macro-and micro level (Greene & Azevedo, 2009) of cognitive and regulation processes as focused on debugging the model. The coding unit was the episode level, which means that coding could be assigned to a single talk turn or alternatively to several consecutive talk turns together, depending on the content of the group's interaction. A cognitive process was identified when students' talk was about problem identification, problem representation, error location, error correction and/or model testing but without explanation (Klahr & Carver, 1988). On the other hand, regulation processes were identified during collaboration when students negotiated and developed problem understanding, planning, enactment, and monitoring and evaluation processes with each other (Winne & Hadwin, 1998). Negotiation and explanation entailed students' discussions of different thoughts and formulation of joint perceptions. Based on the joint framework, each discourse thread in this study was categorized into one of the six problem solving sub-tasks (i.e. micro analysis): *problem identification, problem representation and error location, error correction and model test, problem understanding, planning and enactment, and monitoring and evaluation.* We also identified off-task talk when the students' talk wasn't related to the task. The coding scheme used to identify cognitive and regulation processes with example threads are listed in Table 2. Inter-rater reliability for the macro and micro level of process coding was checked for 10% of the data, resulting in Cohen's kappa values of k= 0.94 and 0.89 respectively, which represents excellent agreement.

Table 2: Regulation and cognitive processes coding scheme along with definitions and representative quotes from transcripts.

Regulation process		Cognitive process	
Description	Examples	Description	Examples
Problem understanding		Problem Identification	
- Read and interpret the task instructions. - Activate previous knowledge of the task - Recognize & explain the problem	S3: "yeah, we'll just round it up to -11. Unless you want to find the dock, because technically we're starting from the dock."	-Read the task instructions without interpretation -Recognize there's a problem (bug) without explanation	S1: "the boat doesn't stop"
Planning and Enactment		Problem representation & Error location	
Determine and describe: -what actions, blocks and resources are needed or what to do next, the group sets a goal for the work to be done; the group sets a task-specific goal	S2: "let's make this an if else and put stop simulation in the else so once it gets there it should stop moving"	- Do or instruct Step or solution without explanation. - Ask for trial and error, guess and check strategies.	S1: "change y position in here")
Monitoring and Evaluating		Error correction & Model-test	
-Provide feedback to each other's ideas or solution -Explain and Analyze components that are responsible for the misbehavior. - Agree with other's solution with explanation -Critique another's suggestion and provide a reason for it.	S1: "We were going to hardcode it to use the equations so that if the distance is changing…", S2: "ooh yeah that's right. Put the update position aside and put set velocity in there"	- Students do not learn what to do after a negative outcome. -Agree or deny another's suggestion without provide a reason for it.	S2: "I put the =, it doesn't work, so I put > and it still doesn't work").

Results

RQ1: How do groups' debugging behaviors relate to their summative learning gains in STEM and CT?

We examined each group performance measures including proportion of total time spent in debugging, number of debugging actions, and mean debugging duration per episode. Also, each group's physics and CT score based on the rubric are presented in Table 3. The correlation between groups' debugging behaviors and their physics and CT model score was tested using the r-values of Spearman correlation. We found a positive correlation between groups proportion of debugging time and both physics and CT model building scores (rs= 0.71, 0.77 respectively). Furthermore, there was a negative correlation between groups' number of debugging actions and both physics and CT model building scores (rs= 0.65, 0.72 respectively). However, these correlations were not significant. However, there was a significant positive correlation between mean debugging duration and both physics and CT model building scores (rs= 0.88, 0.84; $p<0.5$ respectively).

Table 3: Descriptive analysis of debugging behaviors and model-building scores per group

Groups	Total time (seconds)	Duration of debugging (seconds)	No of debugging actions	Mean debugging time (seconds)	Physics score (8)	CT score (10)
G1	1478	1278	17	63	7	9
G2	3602	2744	22	130	8	9
G3	2043	1546	33	47	3	4
G4	3612	1059	45	24	2	2
G5	2520	2288	19	98	6	7
G6	2780	1805	47	38	4	5
Mean (S.D.)	8072.5 (13706)	1786.6 (634.7)	30.5 (13.2)	66.6 (40)	5 (2.3)	6 (2.8)

RQ2: Are there any differences among groups' debugging behaviors?

To answer RQ2, we purposefully selected three groups to understand differences among groups with different performance outcomes in modeling tasks: a) group 2 were the highest performers, nearly successful in the completion of the challenge task, b) group 6 partially completed the task, and c) group 4 had the lowest overall performance scores of groups on the challenge task. A Chi-square test of independence was conducted to analyze the difference between the three groups' debugging behaviors, Table 4. We found significant differences among groups' debugging behaviors ($\chi2$ (10, N=1029) = 1.32, $p < 0.05$). According to Agresti (2007), a standardized residual greater than +2 indicates above-expected observed frequency, whereas a residual smaller than -2 indicates below-expected observed frequency. From Table 4, it can be seen that the frequency of *proportion of debugging time* occurrences were within expectations for all three groups, indicating that they were largely on task.

Group 2 had below-expected frequencies for '*number of debugging actions*'**, while** that was above-expected frequencies for '*mean debugging duration*'. This contrasted sharply with groups 4 and 6 whose observed frequency for '*mean debugging duration*' was below-expectation, while the observed frequency of '*number of debugging actions*' was above-expectation.

Table 4: Chi-square test of independence comparing three groups' debugging behaviors

Debugging behaviors		Group 2	Group 4	Group 6
Proportion of debugging time	Observed Count	76	29	64
	Expected Count	93.3	40.1	61.0
	Std. Residual	-1.8-	-1.8-	.4
Mean debugging duration	Observed Count	130	24	38
	Expected Count	94.2	40.5	61.5
	Std. Residual	3.7	-2.6	-3.0
Number of debugging actions	Observed Count	22	45	47
	Expected Count	40.5	17.4	26.5
	Std. Residual	-2.9	6.6	4.0
Total	Count	228	98	149
	Expected Count	228.0	98.0	149.0

RQ 3: Are there any differences among groups' cognitive and regulation processes of debugging when building computational models?

Overall, all of the groups produced a total of 187 debugging actions. On average, 76% (SD = 28%) of the manifestations of debugging involved discourse threads. The six groups displayed higher frequency of the discussion of cognitive processes than they did of regulation processes. There was, on average, 49% (SD = 15%) of debugging discourse threads involved this category, 37% (SD = 27%) involved regulation, and 13% (SD = 13%) involved off-task talk or no talk. A Chi-square test of independence was conducted to analyze the difference between the three groups' (2, 4, and 6) debugging processes in the distributions of debugging activities. We found significant differences among groups' debugging behaviors ($\chi2$ (10, N=601) = 1.75, $p < 0.05$). From Table 5, we can see that the frequency of occurrences of *proportion of cognitive processes* were within expectation for group 2, while that were above-expected for groups 4 and 6. Only group 4 had above-expected proportion of off-task talk. **Whereas** *the proportion of regulation processes* was above-expectation for group 2, it was below-expectation for groups 4 and 6.

Table 5: Chi-square test of independence comparing groups' processes along with off-task talk

Debugging processes		G2	G4	G6
Cognitive processes	Count	38	62	64
	Expected Count	46.6	46.6	46.6
	Std. Residual	-1.3	2.3	2.6
Regulation processes	Count	54	0	20
	Expected Count	39.3	39.3	39.3
	Std. Residual	2.4	-6.3	-3.1
Off-Task	Count	8	38	16
	Expected Count	14.1	14.1	14.1
	Std. Residual	-1.6	6.3	.5
Total	Count	100	100	100
	Expected Count	100.0	100.0	100.0

We utilized the coding schema in Table 2 to understand how regulation of debugging problems are activated by the students during collaborative model building. Figure 2 shows the results of the proportions of discourse threads for the cognitive and regulation processes sub-categories. A comparison of the three groups revealed that group 4 had higher frequencies of the problem identification threads (42%) than the two other groups. In contrast, group 2 demonstrated a higher percentage of problem understanding threads (15%) than group 6 did (7%). Also, the results show that group 2 displayed a higher percentage of 'planning and enactment' threads (17%) and 'monitoring and evaluating' (20%) than the other groups did. This result indicates that the most successful group frequently discussed the actions that were needed to solve the problem. In other words, they were able to draw their attention to evolving their understanding of the problem to the actions that should be taken to solve the problem. Furthermore, these results reflect that group 2 often monitored the simulation results and provided reflective feedback to their group members, implying that this monitoring and reflecting was a vital component for success in debugging.

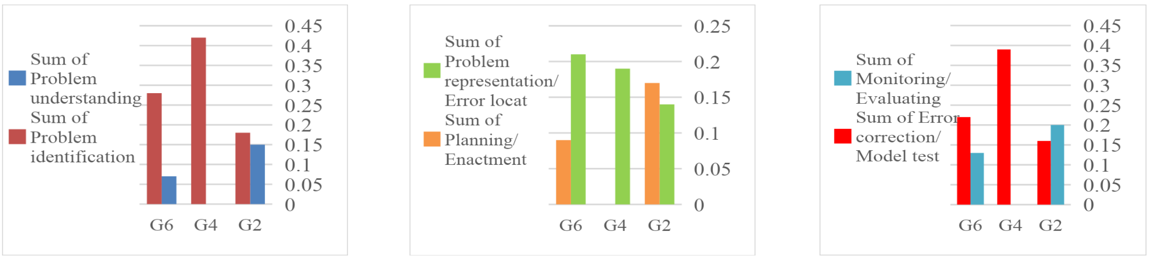

Figure 2. Proportions of discourse threads for cognitive and regulation processes sub-categories per group.

Discussion and implications

Examining 'process' during learning and application of CT practices, and debugging, in particular, in block-based programming has been considered challenging (Grover et al. 2017) and research has been piecemeal, lacking a full picture. The aim of this study was to gain insights into debugging by closely examining cognitive and regulation processes in the context of computational model building in high school physics. Three kinds of debugging behaviors as well as macro-and micro level of debugging processes were observed.

We found that triads who spent more time on debugging errors demonstrated better model building scores in both CT and physics than those that did not. On the other hand, a higher frequency of edit actions to debug errors resulted in lower model building scores. Although this outcome is contrary to that of Liu et al. (2017) who found that those who were faster at debugging (i.e., spent less time) had better task scores, we hypothesize that in our investigations frequent edits likely represented flailing (i.e., trial and error) behaviors.

Our qualitative examination of three purposive naturalistic triads with different performance outcomes shed more light on learners' behaviors. With respect to debugging behaviors, the highest performing triad (Group 2) demonstrated below-expected frequencies for number of debugging actions and above-expectation frequencies for mean debugging duration. Lye and Koh (2018) demonstrated similar results in their examination of programming behaviors (albeit at an individual level). Higher debugging times aligned with low number of debugging actions can be partially attributed to the fact that collaborative actions (group's discussion and explanation) required time and positively affected the triad's ability to analyze and debug their model correctly. Better collaborative regulation involves responding to incorrect model behavior with a monitoring discussion and plan on how to enact a correction. The positive performance of such groups has echoes in our own prior work examining collaboration during model building (Emara, et al., 2017). These behaviors were in contrast with the two other triads. Higher numbers of code edits and short debugging durations could be attributed to multiple attempts to find a solution using trial-and-error methods that are often the result of not focusing on meaningful sub-parts of the problem, or a lack of understanding (Murphy, et al., 2008). Shorter debugging times also indicate fewer deliberative discussions and poorer regulation.

Our conceptual framework combining cognitive and regulation processes makes a valuable contribution to examining how regulation processes in debugging impacts problem-solving behaviors during collaborative computational modeling. We found insights into correlations between students' regulatory processes, problem-solving behaviors, and student performance in computational modeling in a STEM classroom. Our results indicate that overall, the groups paid more attention to identify their problems and then take steps to locate and fix bugs than to the other debugging processes. Only occasionally did they explain and argue about task understanding, planning and executing, and monitoring and evaluating problems that were needed to solve the problem.

This mixed-methods, multiple case studies research involved in-depth analyses and comparison of the characteristics and nuances of modeling behaviors among groups. Our work provides a framework and methodology for understanding learners' modeling behaviors that can be used for further analysis in our own research, and in the field more broadly. A deeper understanding of modeling and problem-solving behaviors as described in this paper can drive the design of intelligent programming and computational modeling environments that afford learners a more supportive, scaffolded learning experience (Grover, et al., 2017). This can be achieved through the use of detectors that are based on findings such as those in this research to provide cognitive and affective support to learners struggling with critical, complex problem-solving tasks such as debugging in an integrated STEM and computing classroom.

References

Agresti, A. (2007). *An introduction to categorical data analysis*. Hoboken, NJ: John Wiley.

Basu, S., McElhaney, K., Grover, S., Harris, C., & Biswas, G. (2018). A Principled Approach to Designing Assessments That Integrate Science and Computational Thinking. *Proceedings of the 13th International Conference of the Learning Sciences (ICLS)*, 384-391.

Chi, M. T. H., Adams, J., Bogusch, E. B., Bruchok, C., Kang, S., Lancaster, M., … Yaghmourian, D. L. (2018). Translating the ICAP Theory of Cognitive Engagement into Practice. *Cognitive Science*.

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Derry, S. J., Pea, R. D., Barron, B., Engle, R. A., Erickson, F., Goldman, R., … Sherin, M. G. (2010). Conducting video research in the learning sciences: Guidance on selection, analysis, technology, and ethics. *The Journal of the Learning Sciences*, *19*(1), 3–53.

Emara, M., Tscholl, M., Dong, Y., & Biswas, G. (2017). Analyzing Students' Collaborative Regulation Behaviors in a Classroom-Integrated Open Ended Learning Environment. *Proceedings of CSCL'17,* 319–326.

Emara, M., Rajendran, R., Biswas, G., Okasha, M., & Elbanna, A. A. (2018). Do Students' Learning Behaviors Differ when they Collaborate in Open-Ended Learning Environments?. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW), 1-19.

Gobert, J., Slotta, J., Clarke, J., Dede, C., Gijlers, H., Saab, N., … Koedinger, K. (2007). Fostering peer collaboration with technology. *Proceedings of the 8th International Conference on Computer Supported Collaborative Learning*, 23–27. International Society of the Learning Sciences.

Goos, M., Galbraith, P., & Renshaw, P. (2002). Socially mediated metacognition: Creating collaborative zones of proximal development in small group problem solving. *Educational Studies in Mathematics*, *49*(2), 193–223.

Greene, J. A., & Azevedo, R. (2009). A macro-level analysis of SRL processes and their relations to the acquisition of a sophisticated mental model of a complex system. *Contemporary Educational Psychology*, *34*(1), 18-

29.
Grover, S., Pea, R. (2018). Computational Thinking: A competency whose time has come. In Sentance, S., Carsten, S., & Barendsen, E. (Eds). *Computer Science Education: Perspectives on teaching and learning,* pp. 19-38.

Grover, S., Hutchins, N., Biswas, G., Snyder, C. & Emara, M. (2019). Examining Synergistic Learning of Physics and Computational Thinking through Collaborative Problem Solving in Computational Modeling. In *Proceedings of AERA, 2019.* Toronto, CA.

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 Transactions on Computing Educ*, 17(3), 14.

Hadwin, A. F., Järvelä, S., & Miller, M. (2011). Self-regulated, co-regulated, and socially shared regulation of learning. Handbook of self-regulation of learning and performance, 30, 65-84.

Hutchins, N., Biswas, G., Maróti, M., Lédeczi, A., Grover, S., Wolf, R., Blair, K., Chin, D., Conlin, L., Basu, S., & McElhaney, K. (2019). C2STEM: a System for Synergistic Learning of Physics and Computational Thinking. *Journal of Science Education and Technology*. https://doi.org/10.1007/s10956-019-09804-9

Klahr, D., & Carver, S. M. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology*, *20*(3), 362–404.

Lin, K.-Y., Yu, K.-C., Hsiao, H.-S., Chu, Y.-H., Chang, Y.-S., & Chien, Y.-H. (2015). Design of an assessment system for collaborative problem solving in STEM education. *Journal of Computers in Education*, *2*(3), 301–322.

Liu, Z., Zhi, R., Hicks, A., & Barnes, T. (2017). Understanding problem solving behavior of 6–8 graders in a debugging game. *Computer Science Education*, *27*(1), 1–29.

Loksa, D., & Ko, A. J. (2016). The Role of Self-Regulation in Programming Problem Solving Process and Success. *Proceedings of the 2016 ACM Conference on International Computing Education Research ICER '16*, 83–91.

Lye, S. Y., & Koh, J. H. L. (2018). Case Studies of Elementary Children's Engagement in Computational Thinking Through Scratch Programming. In *Computational Thinking in the STEM Disciplines* (pp. 227-251). Springer.

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: A review of the literature from an educational perspective. *Computer Science Education*, *18*(2), 67–92.

Murphy, L., Lewandowski, G., McCauley, R., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: the good, the bad, and the quirky--a qualitative analysis of novices' strategies. In *ACM SIGCSE Bulletin* (Vol. 40, No. 1, pp. 163-167). ACM.

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

Peters-Burton, E. E., Cleary, T. J., & Kitsantas, A. (2018). Computational thinking in the context of science and engineering practices: A self-regulated learning approach. In *Digital Technologies: Sustainable Innovations for Improving Teaching and Learning* (pp. 223–240). Springer.

Pintrich, P. R. (2000). The role of goal orientation in self-regulated learning. In *Handbook of self-regulation* (pp. 451–502). Elsevier.

Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. *Computer Supported Collaborative Learning*, 69–97. Springer.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, *18*(2), 351–380.

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education*, *12*(3), 197–212.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725.

Winne, P. H., & Hadwin, A. F. (1998). Studying as self-regulated learning. *Metacognition in Educational Theory and Practice*, *93*, 27–30.

Yen, C.-Z., Wu, P.-H., & Lin, C.-F. (2012). Analysis of experts' and novices' thinking process in program debugging. *International Conference on ICT in Teaching and Learning*, 122–134. Springer.

Acknowledgments
This research is supported by NSF grant #1640199.