Turning Bugs into Learning Opportunities: Understanding Debugging Processes, Perspectives, and Pedagogies

Yasmin Kafai (Chair) University of Pennsylvania, kafai@upenn.edu

Gautam Biswas Vanderbilt University, gautam.biswas@vanderbilt.edu
Nicole Hutchins, Vanderbilt University, nicole.m.hutchins@vanderbilt.edu
Caitlin Snyder, Vanderbilt University, caitlin.r.snyder@vanderbilt.edu
Karen Brennan, Harvard University, karen_brennan@gse.harvard.edu
Paulina Haduong, Harvard University, haduong@g.harvard.edu
Kayla DesPortes, New York University, kayla.desportes@nyu.edu
David DeLiema, University of Minnesota, ddeliema@umn.edu
Oia Walker-van Aalst, UC Berkeley, oia.walker.van.aalst@berkeley.edu
Virginia Flood, UC Berkeley, flood@berkeley.edu
Morgan Fong, UIUC, mmfong2@illinois.edu
Deborah Fields, Utah State University, deborah.fields@usu.edu
Melissa Gresalfi, Vanderbilt University, melissa.gresalfi@vanderbilt.edu
Corey Brady,Vanderbilt University, corey.brady@vanderbilt.edu
Madison Knowe, Vanderbilt University, madison.l.knowe@vanderbilt.edu
Selena Steinberg, Vanderbilt University, selena.k.steinberg@vanderbilt.edu
Diana Franklin, University of Chicago, dmfranklin@uchicago.edu
Donna Eatinger, University of Chicago, dmeatinger@uchicago.edu
Merijke Coenraad, University of Maryland, mcoenraa@umd.edu
Jen Palmer, University of Chicago, jenpalmer@uchicago.edu
David Weintrop, University of Maryland, weintrop@umd.edu
Michelle Wilkerson, UC Berkeley, mwilkers@berkeley.edu
Collette Roberto, UC Berkeley, roberto_c@berkeley.edu
Nicole Bulalacao, UC Berkeley, nbulalacao@berkeley.edu

Joshua Danish (Discussant), Indiana University, jdanish@indiana.edu

Abstract: The design of most learning environments focuses on supporting students in making, constructing, and putting together projects on and off the screen, with much less attention paid to the many issues—problems, bugs, or traps—that students invariably encounter along the way. In this symposium, we present different theoretical and disciplinary perspectives on understanding how learners engage in debugging applications on and off screen, examine learners' mindsets about debugging from middle school to college students and teachers, and present pedagogical approaches that promote strategies for debugging problems, even having learners themselves design problems for others. We contend that learning to identify and fix problems—debug, troubleshoot, or get unstuck—in completing projects provides a productive space in which to explore multiple theoretical perspectives that can contribute to our understanding of learning and teaching critical strategies for dealing with challenges in learning activities and environments.

Overview

The push to implement STEM initiatives in K–12 schools has recently been joined by an equally strong push to re-introduce computer science education under the umbrella of promoting computational thinking and participation for all (Blikstein, 2018). A key computational thinking practice in engineering and computing that is essential, but often overlooked, is debugging: the ability to fix problems in designs that prevent an artifact from functioning as intended. Debugging has always been part of programming. As Papert (1980) noted, "when you learn to program a computer you almost never get it right the first time" (p. 23). However, nearly all the existing research to examine and designs to support debugging have focused on text-based programming languages. As McCauley and colleagues (2008) noted in their comprehensive review of debugging research, it is unclear how findings and strategies developed from these studies apply to block-based programming or even hybrid designs such as electronic textiles. Furthermore, with current efforts to promote integration of computing into STEM

areas, applications such as computer simulations and data visualizations provide new and challenging contexts in which debugging can occur. In this symposium we turn our attention to the many types of debugging—small and large—that students face in making, constructing, and putting together learning projects, be it applications, simulations, or programs, on and off the screen. We see debugging as not only an important *technical* skill that can engage students in a critical application of computational thinking but also a *developmental* skill in which students learn to face challenges and persevere in fixing them using a pedagogical concept that sees failure or problems as providing richer opportunities for learning.

Symposium contributors will examine the role of debugging in the learning process and products as it is situated within a wide range of STEM content areas (e.g., mathematics, science, engineering, and computer science) and learning contexts (e.g., classrooms, afterschool programs, and professional development). Contributions draw from a variety of theoretical traditions (e.g., cognitive, constructivist, and sociocultural theories) and methods (e.g., ethnography, interaction analysis, and interviews). One constant, however, is that presentations focus on debugging as the main topic of study, considering it through three primary lenses which include:

1. Understanding **debugging processes**. How do students learn debugging? What are some of the major challenges? In what kind of processes and strategies do they engage? What kind of mindsets, stances (Fong et al.; Gresalfi et al.) or approaches (Biswas et al; DesPortes; Franklin et al.) do they bring to debugging tasks? How do students engage in debugging physical computing systems which are are on and off the screen (DesPortes; Fields et al.; Franklin et al.)?

2. Comparing how different **disciplinary perspectives** frame students and teachers' actual experiences in identifying and solving bugs. For instance, in computer science it is called debugging (posters by Fong et al.; Brennan et al.; Franklin et al.) while in engineering the focus is on troubleshooting systems (posters by Biswas et al; Franklin et al.; Gresealfi et al.). Activities that involve physical computing such as Arduinos and LEGOs often intertwine these perspectives (posters by DesPortes; Franklin et al.)

3. Considering how **pedagogical approaches** can help learners overcome challenges, in particular those that provide learners—whether teachers or students —with more agency. Here symposium presentations give a wide array of examples: engaging teachers with debugging experiences and materials (Brennan et al.), and providing students with scaffolds (Biswas et al.), opportunities to refactor "inefficient" code (Fong et al.;), mnemonics (Franklin et al.), visualizations (Wilkerson et al.), and tasks that ask students to fix bugs (Gresalfi et al.) and design projects with bugs for others to identify and fix (Fields et al.).

By bringing together these different studies, we contribute to learning sciences research and design in two ways. First, we present the nuanced dimensions through which to understand and operationalize debugging of problems, bugs, or traps: agency (fixing versus designing bugs); context (bugs that are unplanned, unintended, or against hypotheses); and stances (framing, refactoring, reflecting, decomposing). Secondly, the symposium presents a number of practical classroom supports for debugging (building repertoires, providing mnemonics, developing bugs), each of which is actionable, warrants further research, and should be considered alongside the other factors.

After a brief introduction by the chair to the theme and focus of the symposium, each presenter will give a two-minute overview of how they approached debugging in the research or design of learning activities, followed by visits to different poster stations. Our discussant will review overarching themes and address the following research questions: (1) How can we understand debugging in different disciplinary contexts? What are overlapping features? Where are distinctions? (2) In which ways do students deal with challenges and failure? (3) How can teachers learn and provide support to students' debugging efforts? What do teachers need to know about students' challenges? And (4) What are important directions for future research? We will then open up the panel to Q&A with the audience.

Getting unstuck: Designing teacher resources to support conceptual and creative fluency with programming

Karen Brennan and Paulina Haduong, Harvard University

K–12 introductory programming experiences are often highly scaffolded, and it can be challenging for students to transition to open-ended programming activities—activities that necessarily involve moments of failure—later in (and out of) school. Teachers can provide critical support for learners to "get unstuck" from these moments of failure, helping students develop the cognitive, social, and emotional capacities required for conceptually and

creatively complex programming challenges (McCauley et al., 2008). However, teachers, and particularly elementary and middle school teachers, often lack the programming content knowledge, skills, and practices needed to support deeper and more meaningful programming experiences for students (Blikstein, 2018).

Professional development opportunities can cultivate teacher expertise, especially when supported by curricular materials that bridge teachers' professional learning and students' classroom learning. In this poster session, we will present *Getting Unstuck* (https://gettingunstuck.gse.harvard.edu/), our NSF-funded approach to supporting teacher learning through the design and evaluation of (1) an email-based learning experience for teachers to develop conceptual and creative fluency through 21 days of short, daily programming prompts featuring the Scratch programming environment (Haduong & Brennan, 2019) and (2) educative curricular materials for the classroom (Davis, Palincsar, Smith, Arias, & Kademian, 2017), based on the online experience and co-designed with participants.

Getting Unstuck emphasizes three activities to support the development of conceptual and creative fluency with programming: *create*, *share*, and *reflect*. For *create*, learners construct a project in response to a programming prompt that emphasizes a particular computational concept. For example, the prompt "Create a project that controls the appearance of a sprite via the keyboard" invites learners to explore interactivity and events. For *share*, learners post their projects online via the Scratch online community to receive feedback from others. For the initial offering of Getting Unstuck, several thousand projects were shared by 400 participating teachers over the 21 days. In addition to receiving feedback on their own work, sharing facilitated access to others' projects, enabling learners to experience different approaches to problem solving and getting unstuck. For *reflect*, learners respond to questions, either in a digital journal on the project's Scratch community page, about their project-building process, with an emphasis on challenges, failures, and bugs—and how they responded to those challenges. In this session, we will focus on the translation of the *create-share-reflect* activities from our email-based experience to educative curricular materials for 3rd through 5th grade students and their teachers.

Debugging for synergistic learning in learning by modeling environments

Gautam Biswas, Nicole Hutchins, and Caitlin Snyder, Vanderbilt University

Calls to increase authentic engagement in science practices (NGSS, 2013) have highlighted the need for integrating computational modeling in STEM classrooms. Computational modeling is a multifaceted process that enables students to develop critical thinking and problem solving skills. Students must plan, develop, test, and refine their models for use in learning about and applying scientific phenomena (Hutchins et al., 2018, Snyder et al., 2019). We propose that debugging is a key modeling process that requires the tight integration of learning domain and computational thinking (CT) concepts and practices, and we term this integration as *synergistic learning* that simultaneously enhances STEM and CT learning opportunities. Analyzing debugging processes can help us understand how students combine their STEM and CT knowledge to build computational models and identify and correct errors in these models. To evaluate this claim, we analyze computational modeling actions and discourse data during debugging processes to demonstrate the role of debugging on *synergistic learning*.

Our work centers on applications of our Collaborative, Computational STEM (C2STEM) learning environment in high school physics classrooms. C2STEM extends a visual, block-based programming approach by providing domain-specific modeling blocks in the target STEM domains along with associated data processing tools. Following scaffolded, instructional tasks targeting the development of STEM conceptual knowledge (Hutchins et al., 2019), students are tasked with developing computational models that require CT skills (e.g., the programming of a truck to start from rest, speed up to a speed limit, maintain that limit, and then slow down to a stop at a stop sign). In particular, we have analyzed students' model building work (individual and collaborative) through debugging episodes in three key modeling applications: (1) applying *systems thinking* to identify and initialize needed variables and how they interact for computational modeling; (2) *decomposing model behaviors* using conditional logic to determine stopping conditions, and when behaviors of objects change in the environment (e.g., from cruising to slowing down), and (3) *analyzing and interpreting data* to understand and correct model behaviors. Developing debugging practices (e.g., utilizing visual feedback from the simulation and environment tools or evaluating data capture tools based on expected outcomes to identify coding errors) in conjunction with other NGSS science practices also provides key opportunities for students to develop synergistic learning of STEM and CT concepts and practices. We provide our analysis of these processes and discuss implications for future integration of learning-by-modeling in STEM domains.

Debugging with the Arduino: Insights from a lab-based analysis of novices debugging their mistakes

Kayla DesPortes, New York University

Physical computing learning environments can be challenging landscapes for novices to navigate. While the cross-disciplinary nature offers opportunities to leverage the physical nature to integrate computing in unique and meaningful ways, it can be difficult for learners to work across the coding, the electronics, and the materiality of the projects. Due to the complexity, debugging in these environments presents unique hardships as the mistakes across the coding and electronics impacts the functioning and perception of the state of the system and its "correctness". Prior work has demonstrated that even skilled programmers' debugging skills do not necessarily translate to physical computing activities (Booth et al., 2016). While some researchers and developers have chosen to begin to blackbox certain components of either the electronics or code (Searle et al., 2016), the domain of debugging needs to be further analyzed to understand novices approaches and processes, such that we can develop new techniques for assisting in the process.

The work advocates for the need to consider debugging as a central component of physical computing education, while developing insights into how we might accomplish this. The poster examines a real-time analysis of 15 novices' debugging practices while working in a laboratory environment using the Arduino. In the study, 5 of the 15 students completed a think-aloud protocol throughout the process, while the other 10 traversed the learning activities and tasks without thinking aloud. While the Arduino has been noted for the difficulties it might introduce into the learning environment, it still remains one of the most common prototyping devices in secondary and post-secondary educational environments. The study has been reported on in a previous publication (DesPortes & DiSalvo, 2019), and these results expand the contributions with a deeper analysis of the debugging practices and processes inclusive of exploring when and how bugs were introduced across the circuits and code, how long they persisted, how the bugs were diagnosed, and a reflection of what this might mean for how we design scaffolding surrounding debugging.

Debugging by design: Learner and teacher perspectives on making bugs for others

Deborah A. Fields, Utah State University, and Yasmin B. Kafai, University of Pennsylvania

While research on debugging reaches back into the early days of educational computing (e.g., Soloway & Spohrer, 1991), most approaches to teaching debugging tend to focus on succinct, screen-based problems with the objective of finding and fixing one bug in a piece of code (see McCauley et al., 2008). However we see two problematic issues with this type of approach. First, bugs that arise in open-ended design situations are often much more complex, with multiple problematic bugs overlapping and difficult to identify. In situations of physical computing (e.g., robotics, electronic textiles) the situation can become even more complex as bugs may occur on and off screen, and even in ways that stretch across both spaces, creating even more challenging situations for students seeking to isolate, identify, and fix problems. Second, designing such debugging scenarios often takes on a surprisingly instructionist stance where the teacher is in control of choosing and creating bugs. Here we propose a paradigmatic twist by having students intentionally design buggy (rather than functional) computational artifacts.

In this poster, we examine the feasibility of students' debugging by design (DbD for short) for learning and teaching about debugging. We present findings from a classroom study in which students created, exchanged, then solved buggy projects for each other. The study took place over a period of 8 class periods (~55 minutes each) about three-quarters of the way through a larger, 12-week electronic textiles unit (http://exploringcs.org/e-textiles) in a 9th grade high school introductory computing class in the United States. E-textiles involve stitching circuits with conductive thread to connect sensors and actuators to microcontrollers. Making an e-textile involves not only designing functional circuits but also writing code that controls interactions—thus providing myriad opportunities for bugs in crafting the physical artifact and designing the circuits and programming. In this poster we introduce the design of the DbD unit and address the following research questions: (1) What did students gain from this experience? (2) How did participation in DbD shape students' later design experiences in the e-textiles unit?

Data for the study include daily written field note observations, daily teacher reflections, student artifacts (buggy project designs and written reflections), and reflective interviews (n=21 of 25 students) conducted at the end of the unit where students considered the role of DbD in their later designs, collaborations, and problem-solving practices. Analysis primarily consisted of two-step, open coding of transcribed student interviews and reflections (Charmaz, 2014), followed by a comparison with the teacher's daily reflections and recorded observations to provide a fuller picture of the experience and expand on design recommendations. Overall findings included a broader knowledge base of types of problems, better abilities to detect and fix problems, improved

collaborative problem solving with others, and greater confidence in debugging. In the full poster, we explore these findings in detail and address opportunities and challenges that debugging by design provides in student learning.

When features become bugs: Stance-taking around refactoring in a coding classroom

Morgan Fong, University of Illinois Urbana-Champaign, Oia Walker-van Aalst, University of California, Berkeley, Virginia Flood, University of California, Berkeley, and David DeLiema, University of Minnesota

Working solutions to problems are not definitive end points. Because computer programming is a discipline with a long history of valuing the refinement of working solutions toward standards of elegance, efficiency, communicability, and deeper knowledge (e.g., K–12 Computer Science Framework, 2016), code that is technically correct can be treated as buggy. This process of *refactoring* already viable code, which among professionals is "widely recognized as one of the principal techniques" of software development (Demeyer, 2005, p. 1), demands further attention in K-12 CS education spaces to understand the *power-laden* context in which instructors and students *position* (van de Sande & Greeno, 2012) code as "inefficient" or "inelegant," including how students publicly *contest* or *oppose* one another's stances (Dubois, 2007). We thus ask: How (if at all) do instructors and students' efforts to see, promote, and resist refactoring reveal facets of power and inequity in a coding classroom?

To answer this question, we document how 5th-6th grade students and their undergraduate instructors weigh the possibility of refactoring working code in an informal summer CS workshop. We examined a 25-minute stretch of classroom activity in which a hard-coded sequence, copy-and-paste, a series of loops, and one nested loop were explicitly evaluated as routes to the same solution. We transcribed and analyzed through the method of conversation analysis all of the talk, gesture, and action (Goodwin, 2018) in whole-class discourse and at three tables (14 students total), based on video and screen recordings of each table and laptop.

With an eye toward the amplification and attenuation of inequity in collaborative programming spaces (Shah & Lewis, 2019), we describe how the refactoring stance-taking process involved *evaluating* the possibility of revision, *positioning* oneself in support or opposition of revision, and *aligning* or *misaligning* with other prior stances. The study reveals how students enacted power to voice their stance (e.g., "I don't feel like using a for-loop. It's too much work"), misalign or align explicitly with previous stances ("It's actually a lot less work"), and drive action through their stance, including how instructors endorsed (e.g., "I'm shook") or marginalized ("He found copy and paste a hack") different routes. Because it is within these discourse spaces that instructors and students introduce, characterize, promote, and contest disciplinary values about being a coder, educational researchers should carefully understand these social dynamics to foster supportive learning spaces.

WHAT A MESS: Developing strategies to introduce K-8 students to debugging

Diana Franklin, University of Chicago, Donna Eatinger, University of Chicago, Merijke Coenraad, University of Maryland, Jen Palmer, University of Chicago, and David Weintrop, University of Maryland

Debugging can be a foreign and frustrating process for learners as they confront unexpected outcomes - delaying the gratification of seeing their program work as intended. With young children, even relatively simple bugs may derail their engagement with the learning activity. However, the process of debugging can be generative for learners and bugs often expose gaps in understanding and misconceptions providing insight to educators (McCauley, Fitzgerald, Lewandowski, Murphy, Simon, Thomas, & Zander, 2008). When teaching debugging, a central goal is enabling students to choose debugging strategies strategically based on the type of error they identify (Rich, Strickland, Binkowski, & Franklin, 2019). Since one of the most difficult parts of debugging for students is finding the bug in the first place (Fitzgerald, McCauley, Hanks, Murphy, Simon, & Zander, 2010), our work creates mnemonic instructional scaffolds for students to bootstrap the debugging process. For this work, we sought to answer the question: How can we scaffold young learners in debugging Scratch projects? To answer the question, we began by observing students playing Scratch Charades, a game where students act out Scratch scripts, to identify types of errors. These errors were then used to create a debugging mnemonic to help young learners develop debugging skills. The developed mnemonic is WHAT A MESS. It first helps learners identify the issue: **W**hat is the program trying to do, **H**ow did it go wrong, **A**nalyze what happened and **T**hree before me (a common peer-first help-seeking strategy), and then directs learners' attention to common sources of bugs: **A**rguments, **M**issing blocks, **E**xtra blocks, **S**crambled blocks, and **S**ubstituted blocks.

In this poster, we present the WHAT A MESS debugging scaffold and show how it was integrated into a formal Introduction to Debugging unit within Scratch Encore (Franklin et al., 2020), a middle-school intermediate Scratch curriculum. The debugging mnemonic was integrated into early activities alongside learning strategies designed to help students attend to various components of a Scratch program. We created a series of Scratch Encore activities to lead students through noticing a bug, identifying its cause, and fixing the bug with the help of the WHAT A MESS framework. These activities were taught in 25 middle school classrooms (grades 5-8) in a large, urban school district. To understand how learners employed the WHAT A MESS approach, we use automated assessment tools to examine debugged Scratch projects from the students and are analyzing the curricular worksheets for these lessons This poster contributes to our understanding of how to scaffold young learners in developing debugging skills and introduces the WHAT A MESS mnemonic as an accessible introduction to the essential computing practice of debugging.

Debugging data: Diagnosing, evaluating, and repairing data for analysis

Michelle Hoda Wilkerson, Collette Roberto and Nicole Bulalacao, UC Berkeley

Data analysis and visualization is becoming a major focus in computing education (Gould, et al., 2016; Magenheim, 2017). Computational tools and methods have changed how data are collected, explored, and analyzed; this has provided scientists and students unprecedented access to existing datasets that can be adapted for use toward a given line of inquiry (Wallis, Milojevic, Borgman, & Sandoval, 2006). But while there is growing interest in supporting students' analysis and visualization of such datasets, the process of data preparation is often discounted as only a precursor or barrier to meaningful data work (e.g., Kandel, et al., 2011).

In this poster, we build on information visualization (Wickham, 2016) and data science education (Erickson, et al., 2019) literatures to reconceptualize the data preparation process as one of data debugging. Importantly, as debugging highlights the fallibility and social construction of computer programs, data preparation as "debugging" highlights the social and cultural nature of data as scientific text. McCauley and colleagues' (2008) review of debugging strategies highlights a general sequence in which programmers: (1) compare intended and actual program outcomes; (2) learn program structure; (3) evaluate possible sources of breakdown; and (4) repair. We similarly find students engage in data preparation through (1) comparing provided data to intended/expected uses; (2) learning the data's structure; (3) evaluating misalignments; and (4) manipulating the data accordingly through restructuring, merging, filtering, constructing new measures or visualizations, and so on.

This poster will illustrate how students build data debugging repertoires across three studies. In the first, secondary school students (n=14) prepared and analyzed data collected during video game play in order to improve their gameplay strategies. In the second, secondary and community college students (n=10) completed a sequence of task-based interviews in which they were asked to manipulate and analyze provided datasets in order to investigate multivariate relationships, including temporal and spatial relationships, in complex systems. In the third project, middle school students (2 teachers; 5 classes) manipulate public datasets in order to reflect and explore consequential everyday practices—such as analyzing water consumption in light of frequent drought conditions, or analyzing the nutritional content of breakfast routines.

Framing the task: Debugging vs. exploring as opportunities to learn

Melissa Gresalfi, Corey Brady, Madison Knowe, Selena Steinberg, Vanderbilt University

Debugging has been identified as a significant practice of programming. It is both an inherent aspect of programming—that is, figuring out on the fly what something might not be working—and also a separate practice that could possibly be learned and perfected separate from specific programming environments (Lee et al, 2014; Rich et al, 2019). However, we know less about how students learn about debugging, and what kinds of activities and environments influence that learning. Centalr to our definition of debugging is the notion of repair. Specifically, we see debugging as an act of establishing a conversation with code that can be interrogated, refined, and extended iteratively. Who is involved in the conversation might matter--if you are attempting to mediate between someone else's ideas and the code they produced, that sets you up for a different kind of activity (maybe) than if you are mediating between your own ideas and your own code.

In this poster we explore debugging as a conversation by considering the ways the *framing* of a task influences students' programming activity. Frames are people's (often tacit) answers to the question, "What is it that's going on here?" (Goffman, 1974, p. 8), and these expectations influence the ways students respond to curricular tasks and to others in a setting. Specifically, we explore how two intentionally different frames—fixing and exploring—position students to engage similar tasks differently. In the study, which took place in the context of a week-long summer camp about Art and Coding, we introduced students (n=20) to the formal idea of

debugging through a series of complex problems that were embedded in the programming environment with which they were familiar (NetLogo). We asked them to "fix a bug" by considering a client's failed solution to their desired end goal, and to offer "new ideas" by expanding on the client's goal in different ways. Our analysis considers students' approach to these different tasks, and specifically, how being asked to identify a mistake, on the one hand, attunes students to different possibilities and potentials than being asked to explore possible extensions. This analysis contributes to our ongoing understanding of debugging as an interactive accomplishment between person and environment, and specifically helps identify the importance for designers to think about the frames that students bring to bear on their activity as a critical aspect of their engagement.

References

Basu, S., Biswas, G., Sengupta, P., Dickes, A., Kinnebrew, J. S., & Clark, D. (2016). Identifying middle school students' challenges in computational thinking-based science learning. *Research and Practice in Technology Enhanced Learning*, 11(1), 1-35.

Blikstein, P. (2018). *Pre-college computer science education: A survey of the field*. Mountain View, CA: Google LLC.

Booth, T., Stumpf, S., Bird, J., & Jones, S. (2016). Crossed wires: Investigating the problems of end-user developers in a physical computing task. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 3485-3497). ACM.

Brown, A. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences*, *2*(2), 141-178.

Charmaz, K. (2014). *Constructing grounded theory*. Sage publications, Thousand Oaks, CA.

Cooper, S., & Cunningham, S. (2010). Teaching computer science in context. *ACM Inroads*, 1(1), 5–8.

Davis, E. A., Palincsar, A. S., Smith, P. S., Arias, A. M., & Kademian, S. M. (2017). Educative curriculum materials: Uptake, impact, and implications for research and design. *Educational Researcher, 46*(6), 293–304.

Demeyer, S., Van Rysselberghe, F., Gîrba, T., Ratzinger, J., Marinescu, R., Mens, T., et al. (2005). *The LAN-simulation: A Refactoring Teaching Example*. Proceedings of the 2005 8th IWPSE, Lisbon, Portugal.

DesPortes, K., & DiSalvo, B. (2019). Trials and Tribulations of Novices Working with the Arduino. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (pp. 219-227). ACM.

Du Bois, J. W. (2007). The stance triangle. In R. Englebretson (Ed.), *Stancetaking in Discourse* (pp. 139-182). Amsterdam, PA: John Benjamins Publishing Company.

Erickson, T., Wilkerson, M. H., Finzer, W., & Reichsman, F. (2019). Data moves. *Technology Innovations in Statistics Education, 12*(1).

Fitzgerald, S., McCauley, R., Hanks, B., Murphy, L., Simon, B., & Zander, C. (2010). Debugging from the student perspective. IEEE Transactions on Education, 53(3), 390–396.

Franklin, D., Palmer, J., Coenraad, M., Cobian, M., Beck, K., White, M., Anaya, M., Krause, S. & Weintrop, D. (2020). Scratch Encore: The Design and Pilot of a Culturally-Relevant Intermediate Scratch Curriculum. Paper to be presented at the 51st ACM Technical Symposium on Computer Science Education.

Goodwin, C. (2018). *Co-operative action*. New York, NY: Cambridge University Press.

Goffman, E. (1974). Frame analysis: An essay on the organization of experience. Harvard University Press.

Gould, R., Machado, S., Ong, C., Johnson, T., Molyneux, J., Nolen, S., Tangmunarunkit, H., Trusela, L., & Zanontian, L. (2016). Teaching data science to secondary students: The mobilize introduction to data science curriculum. In J. Engel (Ed.), *Promoting understanding of statistics about society: Proceedings of the Roundtable Conference of the International Association of Statistics Education (IASE)*, July 2016, Berlin, Germany.

Haduong, P., & Brennan, K. (2019). *Helping K–12 teachers get unstuck with Scratch: The design of an online professional learning experience*. Paper presented at the 50th ACM Technical Symposium on Computer Science Education (SIGCSE'19), Minneapolis, MN.

Hutchins, N., Biswas, G., et al (2019, in press). C2STEM: A System for Synergistic Learning of Physics and Computational Thinking. *Journal of Science Education and Technology.*

Hutchins, N., Biswas, G., Conlin, L., Emara, M., Grover, S., Basu, S., & McElhaney, K. (2018) Studying Synergistic Learning of Physics and Computational Thinking in a Learning by Modeling Environment. In Yang, J. C. et al. (Eds.). *In Proceedings of the 26th International Conference on Computers in Education (ICCE)* (pp. 153-162). Manila, Philippines,.

K–12 Computer Science Framework. (2016). Retrieved from http://www.k12cs.org.

Kandel, S., Heer, J., Plaisant, C., Kennedy, J., Van Ham, F., Riche, N. H., Weaver, C., Lee, B., Brodbeck, D., & Buono, P. (2011). Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization, 10*(4), 271-288.

Kapur, M. (2008). Productive Failure. *Cognition and Instruction, 26*,(3), 379-424.

Lee, M. J., Bahmani, F., Kwan, I., LaFerte, J., Charters, P., Horvath, A., ... & Long, S. (2014, July). Principles of a debugging-first puzzle game for computing education. In 2014 IEEE symposium on visual languages and human-centric computing (VL/HCC) (pp. 57-64). IEEE.

Magenheim, J. (2017). Data science as a school subject in secondary education from the perspective of computer science education. In *Paderborn Symposium on Data Science Education at School Level 2017: The Collected Extended Abstracts* (p. 95).

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.

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

Papert, S. (1980). *Mindstorms.* New York, NY: Basic Books.

Rich, K. M., Strickland, C., Binkowski, T. A., & Franklin, D. (2019, February). A K-8 Debugging Learning Trajectory Derived from Research Literature. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 745-751). ACM.

Searle, K. A., Tofel-Grehl, C., & Allan, V. (2016). The e-textiles bracelet hack: Bringing making to middle school classrooms. In *Proceedings of the 6th Annual Conference on Creativity and Fabrication in Education* (pp. 107-110). ACM.

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.

Shah, N., & Lewis, C. M. (2019). Amplifying and attenuating inequity in collaborative learning: Toward an analytical framework. *Cognition and Instruction, 37*(4), 423-452.

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. *In Proceedings of the International Conference on Computer Supported Collaborative Learning*, Lyon, France.

Soloway, E., & Spohrer, J.C. (Eds.). (1989). *Studying the novice programmer*. Hillsdale, NJ: Lawrence Erlbaum.

van de Sande, C.C., & Greeno, J.G. (2012). Achieving alignment of perspectival framings in problem-solving discourse. *Journal of the Learning Sciences, 21*(1), 1-44.

Wallis, J. C., Milojevic, S., Borgman, C. L., & Sandoval, W. A. (2006). The special case of scientific data sharing with education. *Proceedings of the American Society for Information Science and Technology, 43*(1), 1-13.

Wickham, H. (2016). *ggplot2: elegant graphics for data analysis.* Springer.

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.