Productive Designs for Successful Failure: Constructionist Perspectives on Supporting Personally Meaningful and Culturally Empowered Learning and Teaching

Luis Morales-Navarro & Yasmin B. Kafai (Chairs)

University of Pennsylvania, Philadelphia, PA, United States, luismn@upenn.edu, kafai@upenn.edu

Karen Brennan, Paulina Haduong, & Varsha Venkatasubramanian

Harvard University, Cambridge, MA, United States, karen_brennan@gse.harvard.edu, haduong@g.harvard.edu, varsha_venkatasubramanian@gse.harvard.edu

Heike Hennig & Tilman Michaeli

Technische Universität München, Munich, Germany, heike.hennig@tum.de, tilman.michaeli@tum.de

David Weintrop

University of Maryland, College Park, MD, United States, weintrop@umd.edu

Jennifer Tsan

WestEd, Chicago, IL, USA, jennifertsan@uchicago.edu

Diana Franklin

University of Chicago, Chicago, IL, USA, dmfranklin@uchicago.edu

Yerika Jimenez & Christina Gardner-Mccune

University of Florida, Gainesville, FL, United States, jimenezyerika@gmail.com, gmccune@ufl.edu

Colin Hennessy Elliott

Drexel University, Philadelphia, PA, United States, ch3457@drexel.edu

Michael Schneider & Jeffrey B. Bush

University of Colorado, Boulder, CO, United States, Michael.J.Schneider@colorado.edu, Jeffrey.Bush@colorado.edu

Deborah A. Fields, Mimi Recker, & Jessie Nixon

Utah State University, Logan, UT, United States, deborah.fields@usu.edu, mimi.recker@usu.edu, jessie.nixon@usu.edu

Francisco Castro & Kayla DesPortes

New York University, New York, NY, United States, francisco.castro@nyu.edu, kayla.desportes@nyu.edu

Mike Tissenbaum, Casey Smith, Ashita Bawankule, & David Hopping

University of Illinois, Urbana-Champaign, IL, United States, miketiss@illinois.edu, cjsmith0@illinois.edu, anb8@illinois.edu, davidah3@illinois.edu

Nathan Holbert, Isabel Correa, Blake Danzig, David Zikovitz, & Paulo Blikstein

Teachers College, Columbia University, New York, NY, United States, holbert@tc.columbia.edu, mic2130@tc.columbia.edu, bpd2119@tc.columbia.edu, dmz2117@tc.columbia.edu, paulob@tc.columbia.edu

Matthew Berland (Discussant)

University of Wisconsin-Madison, Madison, WI, United States, mberland@wisc.edu

In education, we cherish success and fear failure. But not every learning experience is a success right away, in fact failing and making mistakes and then learning from them is the norm rather than the exception. In this symposium, we present different perspectives on how failure can contribute to constructionist learning and teaching, examining how learners identify and address failures in their designs and how distinct approaches to failure can support learners in collaboratively creating personally meaningful projects. Taking a holistic approach to failure—that incorporates cognitive, social, and affective factors—we argue that failure should play a key role in constructionism and present different perspectives for finding a more productive stance that turns failures into rich opportunities for constructionist learning and teaching.

Keywords and Phrases: Failure, Bugs, Constructionism, Debugging, Computing Education

1 SYMPOSIUM OVERVIEW

While traditional instructionist approaches to learning frame failure and errors as undesirable, when making constructionist projects encountering failure is nearly inevitable. As Papert (1980) remarks, “errors benefit us because they lead us to study what happened, to understand what went wrong, and, through understanding, to fix it.” Recent scholarship has shifted our perspectives on how errors, mistakes, and failure can be beneficial for learning in open ended activities when designers holistically attend to the learner’s emotional, motivational, social and cognitive aspects of encountering failure (Zhang & Fiorella, 2023; DeLiema et al., 2022). These perspectives complement earlier work that shows that early failure in well-defined learning activities can be beneficial for later student success (Kapur, 2016). Within constructionism, recent efforts on situating failure have taken on a more agentic role highlighting the importance of failure from promoting different strategies for getting unstuck (Haduong & Brennan, 2018) to having students design failure artifacts for learning (Fields et al., 2021). We build on contemporary research on failure and learning—that sees learning from failure as a holistic process that involves cognitive, social, and affective factors—by presenting a range of perspectives on failure in constructionist learning.

Participants will examine how to scaffold students in debugging Scratch projects, how teachers can support students debugging of open-ended projects, how learners encounter failure and design failure artifacts, and how tools can support debugging. Brennan and colleagues examine the implementation of a curriculum designed to support students when they encounter failure in self-directed programming projects. Franklin and colleagues present a mnemonic strategy designed to help novices identify and resolve bugs they encounter while creating projects on Scratch. Jiménez and Gardner-McCune examine students' productive and unproductive debugging behaviors on Scratch. Hennig & Michaeli investigate different types of feedback provided by teachers when students request help with debugging. Hennessy Elliott and colleagues examine how teachers support students in debugging physical computing systems. Morales-Navarro and colleagues analyze student reflections and the different types of bugs they engaged when creating personally relevant buggy physical computing projects for their peers to solve. Castro and DesPortes examine how youth use their bodies, practices, and observations to address failure in dance and computing projects. Fields and colleagues explore the role of failure outside the classroom in a remote Thai village’s development of a sustainable water management system. Tissenbaum and colleagues present a collaborative debugging at a distance tool for physical computing projects. Schneider presents a tool for students to safely explore and test physical computing projects while debugging. The symposium is organized in four sections: (1) the chairs will introduce the topic and then each presenter will give a teaser about their work (~10 min); (2) the first half of

the presenters will have 15 minutes to share their work using posters placed around the room, followed by the second half of presenters (15 minutes)—this arrangement will give the audience and presenters time to see each other's posters; (3) our discussant Matthew Berland will synthesize and reflect on findings (10 minutes) followed by (4) a Q&A with audience and presenters (~10 min).

2 PRESENTED POSTERS

2.1 Self-Directed Programming Projects in the Elementary Classroom: Learning and "Unlearning" (Brennan, Haduong, & Venkatasubramanian)

Though elementary schools are increasingly offering introductory computer science learning experiences, it remains challenging to structure opportunities that enable young learners to develop and implement self-directed projects. Part of the challenge resides in helping students manage the inevitable moments of getting stuck and encountering failure. These experiences of failure in self-directed programming projects invite students and teachers to learn new strategies and "unlearn" aspects of traditional school culture (Cochran-Smith, 2003). To help students engage in self-directed programming projects involving encounters with failure, our research group developed *Getting Unstuck*, a 10-module intermediate Scratch curriculum. The curriculum reimagines the classroom as a design studio: a learning environment in which students are invited to *explore*, *create*, *share*, and *reflect* through self-directed programming projects. In a pilot study of the curriculum conducted with six 3rd, 4th, and 5th grade teachers and their students, we sought to understand what was challenging for students and teachers as students pursued self-directed projects. Through thematic analysis of pilot teachers' reflective journals, students' design journals, interviews with the teachers, and classroom observations, we identified dimensions of learning and "unlearning" in which the students engaged across the design studio activities of exploring, creating, sharing, and reflecting. As students *explored*, students and teachers reimagined their relationship to examples and explicit instruction. As students *created*, students and teachers developed new ways of thinking about remixing and the role of tenacity. As students *shared*, teachers and students learned how to trade perfectionism for the benefits of sharing works-in-progress and how to be supportive digital citizens. As students *reflected*, teachers and students explored the benefits of contemplative practice as more than external accountability. We hope this work underscores the power and potential of supporting self-direction in the classroom. Although this shift in learning culture is replete with challenges, viewing moments of failure as opportunities for development underscores the possibilities of this approach.

2.2 Investigating Teacher Feedback in Debugging using a Video Vignette Approach (Hennig & Michaeli)

Debugging is a core issue in K-12 computing education, challenging students and teachers alike. For students, finding and fixing programming errors is a significant challenge, especially when working in open-ended, constructionist learning environments. For novice programmers, the naturally high number of errors and lack of strategies for dealing with them can lead to frustration. Typically, the students ask the teacher to help them individually with their programming problems (Michaeli & Romeike, 2019). The teacher's goal is to empower students to find and fix the current and similar future bugs themselves while fostering a positive error culture in the classroom. To do this, they need to understand the problem, diagnose why the student is stuck, and choose an appropriate intervention in a very short time. However, there is a lack of research on effective teacher feedback for debugging situations in the classroom (Hennig & Michaeli, 2022). As a first step in addressing this issue, we conducted a study investigating teachers' diagnostic and intervention skills. Using a video vignette approach, we confronted experienced computer science teachers with scripted videos of typical classroom debugging situations: Two students are pair programming and seek

help from the teacher. We asked the teachers for their diagnosis of the student's problem and their reaction using an open-ended questionnaire. This allows a realistic simulation of typical classroom situations, as situational features such as student questions, discussion within the programming pair, and the students' code are available to the teacher. Initial results show that the aspects considered in the teachers' diagnosis are diverse. The proposed interventions are primarily aimed at guiding students to solve the specific problem and only rarely address larger issues, such as how failure can be used for learning.

2.3 **Buggy projects as Objects-to-think-with (Morales-Navarro, Fields & Kafai)**

Supporting students in learning debugging often involves instructionist approaches such as teaching them techniques, following checklists, or having them fix teacher—or researcher—designed programs. Instead, our approach to debugging in Debugging by Design (DbD) builds on constructionism by having learners create personally relevant, buggy applications for their peers. Prior studies on DbD show that creating buggy e-textile projects may support students in identifying and solving bugs while attending to the emotional and motivational toll that debugging often takes (e.g., Fields et al., 2021). Beyond these attitudinal gains, in this study we sought to understand what students *learned* in designing bugs: What kind of bugs did students create and what does this reflect about their learning and understanding of computing? We investigate the role of buggy projects as objects-to-think-with, “in which there is an intersection of cultural presence, embedded knowledge, and the possibility for personal identification” (Papert, 1980, p. 24) by analyzing the different types of bugs that students designed for each other to solve and their reflections on the design process. In Spring 2019, an experienced teacher at a metropolitan U.S. high school implemented DbD in a classroom with 11 girls and 14 boys. Drawing on observations, photographs, code, and students’ initial designs and plans, we systematically analyzed students’ buggy designs, identifying, and classifying each bug. Overall, 12 groups of students designed a total of 185 bugs, which we classified into 30 categories, from basic syntax (capitalization, punctuation, spelling) to sophisticated logic and design errors. Our analysis demonstrates how students explored the intricacies and interrelatedness of the physical computing system (PCS), from the organization of code structure to multiple types of variables, logic statements, function calls, and conflicts between the physical and coded aspects of projects. In the poster, we consider how this bug-motivated exploration of PCS fulfilled the goals of Papert’s objects-to-think-with.

2.4 **WHAT A MESS: Scaffolding Debugging in Scratch (Franklin, Tsan, & Weintrop)**

Debugging is an essential skill for building and creating with technology. Fixing errors and resolving challenges that emerge while programming is necessary to create functioning software and can also serve as a powerful opportunity for learning (McCauley et al., 2008). Despite the critical role it plays in programming, the practice of debugging is rarely the focus of instruction or explicitly scaffolded in learning environments, instead, learners are often left to their own devices to discover and develop productive debugging strategies. Research on novices learning to debug has identified the conceptual underpinnings and generative practices related to learning to debug (Rich et al., 2019). Informed by this research, we sought to understand how to support novice Scratch programmers in learning productive and effective debugging strategies. Towards this end, we developed the mnemonic: WHAT A MESS! to help novices identify and resolve bugs they encounter. 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 approach and show how it has been incorporated into Scratch Encore (Franklin et al., 2020), a culturally responsive middle school computer science curriculum. This includes the specifics of each element of the mnemonic along with scaffolds developed to help learners employ the various concepts/practices as they work to author personalized and personally meaningful Scratch programs.

2.5 Towards Understanding Students' Debugging Behaviors in Scratch (Jiménez & Gardner-McCune)

Debugging is a complex cognitive process that requires students to learn how to apply different programming skills, such as program comprehension, reading, writing, tracing, predicting, and modifying programs (McCauley et al., 2008). Learning these skills can be challenging for novice students and demotivate them if they are not carefully supported throughout the process. Motivated by a lack of studies focused on in-depth analysis of elementary students' debugging in block-based programming environments, we designed a 4-week online debugging intervention that explicitly taught elementary students the debugging process, as well as how to read, write, trace, predict, modify, and debug in Scratch 3.0 using reciprocal teaching. At the end of the summer camp, students participated in a debugging study where they solved tasks focused on common bugs, they encountered during the summer camp. Through video and constant comparative analysis with ten elementary students, we found that they exhibited both productive and unproductive debugging behaviors while debugging buggy code in Scratch. The productive behaviors students exhibited were found to be directly correlated with the debugging skills that were taught during the four-week debugging intervention. These debugging skills included: 1) code reading before starting the debugging process, 2) testing the code before debugging, 3) iterative testing, 4) tracing through the code while debugging, and 5) talking out loud while debugging. In contrast, unproductive debugging behaviors are defined as actions taken by students that do not lead to successful debugging of code. We found that elementary students experienced attitudinal (e.g., self-doubt, hesitation) and procedural (reducing code complexity) debugging behaviors while debugging in Scratch. Our findings suggest that explicitly teaching debugging skills to elementary students can be effective in improving their ability to debug code, but continued reinforcement and support may be necessary to ensure the skills are retained and utilized effectively.

2.6 Facilitating Failure: How Teachers Support Learning Through Debugging with Physical Computing (Hennessy Elliott, Nixon, Bush, & Recker)

Engaging in debugging tasks with physical computing systems (PCS) illuminates secondary students' emerging understandings about computing, and creates important moments for their teachers to support them encountering instances of failure (DeLeima, et al., 2022). When deciding how to provide guidance, teachers wrestle with uncertainty about what issues students need to resolve and how to best resolve them (Hennessy Elliott, et al., 2022). This paper presents analyses of middle school teachers' interactional work as they support students assembling and programming a sensor-based PCS. The PCS becomes an "object to think with" (Papert, 1980), in the context of instructional activities designed to help students develop understanding of physical phenomena. Using video data of seven teachers over three years, we analyzed debugging moments where teachers worked directly with students encountering issues. Analyses visualized the variety of approaches teachers took on two dimensions capturing debugging pedagogy: an ***understanding dimension***–which captures whether a teacher's approach privileges of their own or the student's understanding–and the ***process/product dimension***–which captures whether a teacher's approach is oriented to supporting students in debugging or in just getting their PCS to work. Teachers' orientation on the two dimensions were not always consistent nor correlated across debugging episodes. Teachers used a large variety of orientations in relation to the kinds of questions and problems students articulated. We posit that these decisions were influenced by teachers' read of students' emotional, cognitive, and social states, as well as other contextual factors. The teacher's role in facilitating student engagement with failure in the context of physical computing was situated and improvisational, relying on their ability to not only figure out bugs in the PCS but also the students' orientations to its failure. This has implications for how professional learning is designed to help teachers support students as emerging debuggers of PCS.

2.7 Exploring with the Body: Situating Failure and Iteration While Learning Dance and Computing (Castro & DesPortes)

Dance and computing learning experiences bridge the digital and physical world to create ways for learners to engage in a variety of practices for thinking, exploration, and learning in which the body and its context are integral. Learners engage with their own dance cultures and identities as they use body positioning and movement to learn and communicate (Castro et al., 2022; Leonard et al., 2021). We explore how goal setting, failure, and iterations towards successes are navigated as learners build dance videos with coded animations using danceON. danceON is an open-access, web-based, creative coding environment that embeds pose detection models and enables learners to upload dance videos over which they can code movement-responsive virtual animations and engage authentically with dance (Payne et al., 2021). In a 15-week remote study, six learners from STEM From Dance used danceON to create dance computing artifacts within individual and group activities, while documenting their design processes through blogs. We analyzed learners' blogs and recording transcripts from the sessions by identifying instances of how learners used their bodies, practices, and observations of danceON behavior to think through misalignments between code and animation behaviors, the limitations of pose detection algorithms in representing their bodies and movements, and the impacts of these misalignments and limitations on the dance computing performances they created. We found that learners iteratively redefined goals throughout the development of their dance computing artifacts (e.g., "happy mistakes") as they negotiated and navigated where "failures" exist, such as in learner's actions or within the limitations of danceON. We demonstrate how learners transformed "failures" into meaningful and productive learning experiences within open ended creative computing environments.

2.8 The Dams Blew Out: A Thai Village's Collective and Constructionist Response to Failure (Fields, Morales-Navarro, Blikstein)

In this poster we share a provocative story about a village's collective response to the failure of months of manual labor in creating a sustainable water management system. Ban Samkha is a rural village of about 600 people, set amongst the hills and forests of Northern Thailand. The area faced annual water shortages due to a lengthy dry season and the lack of any water infrastructure to contain precipitation from the wet season. In March 2546 (2003 in Gregorian years), inspired by a constructionism that validated their collective learner agency and emphasized learning by doing (Fields et al., 2022), the village began to create a system of "check dams", using trees, rocks, and leaves to create small drainage blockages on the mountain near the village to drive water into the ground. Within three months they had built 20 check dams. However, because villagers had built dams at the bottom of the mountain, all the dams blew out from the force of the water during the rainy season. Part of a larger study on the village and constructionism in Thailand (Fields et al., 2022), for this analysis we applied an iterative case study approach to the village's 25-year learning journey, encompassing 37 hours of interviews and 350+ pages of scanned project documentation (mostly from village records and writings) collected across four site visits. We share how in the face of utter failure, the village engaged in collective reflection and iteration, sought expertise and resources (from satellite maps to identify natural water drainage to used chainsaws and donations of cement for stronger dams higher up the mountain), and over 10 years created a system of 20,000+ check dams. This case expands our understanding of constructionist approaches to failure, with both every day and sophisticated technology, for deeply consequential learning.

2.9 Connected Spaces: Technological Infrastructure for Collaborative Debugging at a Distance (Tissenbaum, Smith, Bawankule, Hopping, Holbert, Correa, Danzig, & Zikovitz)

Connected Spaces (C/S) is a technology framework and suite of tools that supports collaborative debugging for geographically distributed middle and high school makers. The goal of Connected Spaces is to benefit learners who are traditionally underrepresented in STEM+C career pathways by connecting them to

like-minded peers and offering new opportunities for them to develop their maker identities and sense of belonging within a larger maker community. Three technologies—the dashboard, help button, and REACH projector—work in tandem to support students in help-seeking, finding collaborators, and getting “unstuck”. The dashboard is an ambient display across makerspaces where learners showcase their skills and their availability to help. Help seekers use the dashboard to find peer support or collaborators. The design aims to support makers' identity building and sense of community by enabling learners to customize their individual representations (e.g., affinities, works-in-progress, etc.) on the dashboard. The HELP button is designed to ease the initial request for help, providing a simple physical interface for selecting a need and broadcasting the request across the connected makerspaces. The REACH projector supports collaborative debugging as a two-way communicator that combines a projector and a camera in an overhead display. Makers put a work-in-progress they need help with under REACH and it is projected into their collaborator's REACH space. Anything the collaborator puts under their REACH is simultaneously projected at the help seeker’s REACH, allowing real-time non-collocated debugging around physical objects. In this poster session, we demonstrate these technologies and share our initial findings from multiple implementations across makerspaces in Illinois and New York that led to the current design. Including an approach that combined coding students' week-by-week of their dashboard personas with one-on-one interviews to understand its impact on their computational identities and applying the Divergent Collaboration Learning Methods (Tissenbaum et al., 2017) framework to understand students' boundary spanning collaborative practices.

2.10 Circuit Check: Enabling Students to Safely Explore while Debugging E-textiles (Schneider)

E-textiles weave computational and electronic components into fabric materials, enabling students to create projects that can sense and interact with the physical world. While this combination of hardware and software can provide an engaging medium for creative learning, it also forms a more challenging space for debugging. Debugging is a known challenge in software engineering (McCauley et al., 2008), but it can be even more challenging in physical computing systems, like e-textiles, where students not only have to contend with errors in their software, but also in their circuit’s hardware and even the connections between the two (Booth et al., 2016). Early student attempts at debugging often follow a trial-and-error approach that consists of guessing at a potential cause, making a quick change to the system, and then hoping everything now works. Unfortunately, this approach is not very forgiving of mistakes where hasty changes can create new bugs instead of solving the original one. While making mistakes is a natural part of debugging, they shouldn’t be so costly for students to make. I designed an interactive debugging tool, Circuit Check, to provide a safer environment for debugging. Unlike traditional debugging techniques, like print statements and commenting out code chunks, which require students to add or remove specific lines of code, Circuit Check enables students to observe sensors and test actuators without any code modifications. This enables students to make incorrect guesses and safely explore their hardware while debugging. I have evaluated Circuit Check’s debugging features through classroom observations of high school teachers and their students who used Circuit Check to debug e-textile projects. Through analysis of video recordings of classroom interactions and teacher interviews, I have found that Circuit Check’s unique design and debugging features support not only students in debugging on their own, but also in collaboratively debugging with their teacher and peers.

REFERENCES

Booth, T., Stumpf, S., Bird, J., & Jones, S. (2016, May). 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).

Castro, F. E. V., DesPortes, K., Payne, W., Bergner, Y., & McDermott, K. (2022). AI + Dance: Co-Designing Culturally Sustaining Curricular Resources for AI and Ethics Education Through Artistic Computing. Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2, 26–27. https://doi.org/10.1145/3501709.3544275

Cochran-Smith, M. (2003). Learning and unlearning: The education of teacher educators. Teaching and Teacher Education, 19(1), 5-28. https://doi.org/https://doi.org/10.1016/S0742-051X(02)00091-4

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. In Computer Science Education (Vol. 18, Issue 2, pp. 67–92). https://doi.org/10.1080/08993400802114581

DeLiema, D., Kwon, Y. A., Chisholm, A., Williams, I., Dahn, M., Flood, V. J., ... & Steen, F. F. (2022). A Multi-dimensional Framework for Documenting Students' Heterogeneous Experiences with Programming Bugs. Cognition and Instruction, 1-43.

Fields, D. A., Kafai, Y. B., Morales-Navarro, L., & Walker, J. T. (2021). Debugging by design: A constructionist approach to high school students' crafting and coding of electronic textiles as failure artefacts. British Journal of Educational Technology, 52(3), 1078-1092.

Fields, D. A., Morales-Navarro, L., & Blikstein, P. (2022). "The village that learns": a learning journey across intraventions and domains over two decades in a rural Thai community. Mind, Culture, and Activity, 29(3), 191-214.

Franklin, D., Weintrop, D., Palmer, J., Coenraad, M., Cobian, M., Beck, K., ... & Crenshaw, Z. (2020). Scratch Encore: The design and pilot of a culturally-relevant intermediate Scratch curriculum. In Proceedings of the 51st ACM technical symposium on computer science education (pp. 794-800).

Haduong, P., & Brennan, K. (2019, February). Helping K--12 Teachers Get Unstuck with Scratch: The Design of an Online Professional Learning Experience. In Proceedings of the 50th ACM technical symposium on computer science education (pp. 1095-1101).

Hennig, H., & Michaeli, T. (2022). Investigating Teachers' Diagnostic and Intervention Skills in Debugging. In Proceedings of the 17th Workshop in Primary and Secondary Computing Education (pp. 1-2).

Kapur, M. (2008). Productive failure. Cognition and instruction, 26(3), 379-424.

Leonard, A. E., Daily, S. B., Jörg, S., & Babu, S. V. (2021). Coding moves: Design and research of teaching computational thinking through dance choreography and virtual interactions. Journal of Research on Technology in Education, 53(2), 159-177.

Michaeli, T., & Romeike, R. (2019). Current status and perspectives of debugging in the k12 classroom: A qualitative study. In 2019 ieee global engineering education conference (educon) (pp. 1030-1038). IEEE.

Papert, S (1980). Mindstorms. Basic Books.

Payne, W. C., Bergner, Y., West, M. E., Charp, C., Shapiro, R. B. B., Szafir, D. A., Taylor, E. V., & DesPortes, K. (2021). danceON: Culturally Responsive Creative Computing. Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, 1–16.

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.

Tissenbaum, M., Berland, M., & Lyons, L. (2017). DCLM framework: Understanding collaboration in open-ended tabletop learning environments. International Journal of Computer-Supported Collaborative Learning, 12(1), 35–64.

Zhang, Q., & Fiorella, L. (2023). An integrated model of learning from errors. Educational Psychologist, 1-17.