RETHINKING DEBUGGING AS PRODUCTIVE FAILURE FOR CS EDUCATION

Yasmin Kafai (Moderator)
University of Pennsylvania
Philadelphia, PA
kafai@upenn.edu

David DeLiema
University of California, Berkeley
Berkeley, CA
deliema@berkeley.edu

Deborah Fields
Utah State University
Logan, UT
Deborah.Fields@usu.edu

Gary Lewandowski
Xavier University
Cincinnati, OH
lewandow@xavier.edu

Colleen Lewis
Harvey Mudd College
Claremont, CA
lewis@cs.hmc.edu

CCS Concepts
Computing Education

Keywords
Computing education; computational thinking; debugging

ABSTRACT

Computational thinking has become the calling card for re-introducing coding into schools. While much attention has focused on how students engage in designing systems, applications, and other computational artifacts as a measure of success for computational thinking, far fewer efforts have focused on what goes into remediating problems in designing systems and interactions because learners invariably make mistakes that need fixing—or debugging. In this panel, we examine the often overlooked practice of debugging that presents significant learning challenges (and opportunities) to students in completing assignments and instructional challenges to teachers in helping students to succeed in their classrooms. The panel participants will review what we know and don't know about debugging, discuss ways to conceptualize and study debugging, and present instructional approaches for helping teachers and students to engage productively in debugging situations.

INTRODUCTION

Debugging has always been an essential part of programming. As Papert [9] noted, "when you learn to program a computer you almost never get it right the first time" (p. 23). Thus, computation always involves the practice of continually diagnosing and fixing problems. Looking at research on debugging from the early days of educational computing, we know a lot about novice programmers' bug types, and their processes and problems with identifying bugs [8]. Beyond mere syntax issues like misspellings or typos, novices' bugs are also semantic in nature, dealing with more thorny issues of errors in run-time or logic design, thereby reflecting their underlying understanding of computation itself. More recent pedagogical approaches offer new ways to reframe such debugging situations. Instead of posing a challenge, debugging can offer opportunities for "productive failure," or the counterintuitive notion that failure precedes later success in learning [3]. Today's extensive research on this concept highlight the role of multiple representations and solutions, activation of prior knowledge, and nature of peer support, in order to identify which dimensions are most productive for different students and conditions [4,5]. This panel will present different perspectives on debugging, starting with an overview of what we know about student challenges and instructional approaches (Lewandowski), followed by research on students and teachers approaches to debugging (DeLiema), and instructional approaches to debugging in undergraduate education (Lewis) and high school classrooms (Fields).

1. GARY LEWANDOWSKI

We are fortunate to have useful literature on debugging that dates back to the 70s and continues to have relevance for understanding the challenges students face and ideas for approaching instruction. Generally speaking, we understand that bugs occur most typically when students experience a cognitive breakdown in skills, rules, or knowledge. The most prevalent kinds of bugs are missing or malformed plan executions, sometimes due to fragile language skills, and other times what Knuth describes as “algorithm awry.” A particular challenge for students in debugging is the ability to think about the overall system that they are building, which makes it more difficult to locate their bugs. Thus some of the suggested instructional approaches are guides to help students pursue either forward or backward reasoning about the bugs. The good news is that there is evidence that when taught a debugging technique, students will use it! Productive failure is likely to fit into the existing literature as a guiding principle that encourages recognition that bugs are a part of the programming process that can lead us to insight on the problem being solved and better (correct) solutions.

Gary Lewandowski is a Professor of Computer Science at Xavier University where he experiences the joys of failure and debugging on a daily basis while teaching computer science and human-centered making, and while serving as Associate Dean. His research interests include debugging processes of students as part of a larger interest in the assets students bring to their study of computer science.

2. DAVID DELIEMA

When teachers, researchers, and parents talk about children handling failure in the learning process in *productive* ways, what might they mean? Blending prior theoretical and empirical research on failure and debugging [3,10], learning conjectures from a multi-year design-research project [1], and micro-longitudinal analyses of a 6th grader’s approach to debugging computer code over two weeks, I propose five characteristics of how students handle failure that each mark a unique facet of productivity: whether (1) problems get fixed, (2) students learn to avoid specific (recurring) problems, (3) students know and learn strategies for debugging novel problems, (4) students drive the process of debugging problems, and (5) students believe they can debug problems successfully. I argue that newcomers to a discipline should foreground and background subsets of these characteristics in different configurations at different points for focused practice. I make this argument by drawing on data that triangulates participation during coding (video, audio, and screen recordings of teacher-student and peer-to-peer classroom interactions), artifacts produced along the way (coding journals, post-it notes, bug tickets, artwork, and written artist statements), and reflections on learning (interviews about debugging and art-making). This study sets a foundation for research on particular characteristics of the social and material contexts that shape how students new to a discipline such as computer programming approach failure.

David DeLiema is a Postdoctoral Researcher in the Graduate School of Education at UC Berkeley where he studies students’ experiences with failure, embodied learning, and epistemic cognition.

3. DEBORAH FIELDS

I argue for a broad view of debugging since it can involve important thinking processes. I am particularly aware of these thinking processes in the context of physically embodied projects such as electronic textiles. E-textiles involve making fabric-based projects by sewing sensors and microcontrollers using conductive thread to make computational circuits. Intersections of crafting, circuitry, and computation make tracing root causes tricky, demanding strategic debugging and revisions productive for learning [7]. In working with hundreds of students creating e-textiles over the past decade we have noted beyond identifying syntax errors, debugging can involve problem-solving, iterative testing, isolating problems, hypothesis testing, and even working with an intended audience for one’s project [2]. In addition to arguing for an expansive consideration of debugging, I also argue that we should help students become aware of the thinking skills they use in debugging projects. We have explored using reflective portfolios for this purpose, inviting students to share things that went wrong and how they resolved those problems. Meta-reflection on debugging has strong potential for helping students’ develop conscious strategies for debugging.

Deborah Fields is an Associate Research Professor at Utah State University, where she investigates student learning through making creative computational artifacts, studying relationships between design, personal relevance, and learning.

4. COLLEEN LEWIS

As a CS education researcher, I hypothesize that debugging is particularly challenging to teach because students over classify bugs as “silly mistakes” and thus disregard the importance of their debugging practice [6]. If a student has enough expertise to *find* the bug, this expertise is likely sufficient to *fix*. That is, once you *find* a bug, it is often trivial to *fix* it. If a bug appears to be trivial, the process of finding the bug may also be dismissed as trivial. However, it is in this practice of debugging where students are building the skills to debug programs. As a CS educator, I have attempted to support my students in developing their debugging skills by giving students assignments that require debugging provided code and by embedding debugging advice where they will come across it when they need it (i.e., as a comment within provided test cases and on a poster in the computer labs csteachingtips.org/tips-for-tutors).

Colleen Lewis is the McGregor-Girand Associate Professor of computer science at Harvey Mudd College. At UC Berkeley, Lewis completed a PhD in science and mathematics education and an MS in computer science. Her research seeks to identify effective teaching practices for creating equitable learning spaces where all students have the opportunity to learn. Lewis curates CSTeachingTips.org, a NSF-sponsored project for disseminating effective computer science teaching practices.

REFERENCES

[1] David DeLiema. 2017. Co-constructed failure narratives in mathematics tutoring. *Instructional Science, 45*, 6 (Dec. 2017), 709-735.

[2] Gayithri Jayathirtha, Deborah A. Fields, and Yasmin B. Kafai. 2018. Computational concepts, practices, and collaboration in high school students' debugging electronic textile projects. In *Proceedings of International Conference on Computational Thinking Education* (CTE '18), The Education University of Hong Kong, Hong Kong, China.

[3] Manu Kapur. 2016. Examining productive failure, productive success, unproductive failure, and unproductive success in learning. *Educational Psychologist, 51*, 2 (April 2016), 289–299.

[4] Manu Kapur and Katherine Bielaczyc. 2012. Designing for productive failure. *Journal of the Learning Sciences*, *21*, 1 (June 2011), 45-83.

[5] Manu Kapur and Nikol. Rummel. 2012. Productive failure in learning from generation and invention activities. *Instructional Science*, *40*(4), (July 2012) 645-650.

[6] Colleen M. Lewis. 2012. The Importance of Students' Attention to Program State: A Case Study of Debugging Behavior. In *Proceedings of the International Computer Science Education Research Workshop (ICER '12)*. ACM, Auckland, NZ, 127-134.

[7] Breanne K. Litts, Yasmin B. Kafai, Kristin A. Searle, and Emily Dieckmeyer. 2016. Perceptions of productive failure in design projects: High school students' challenges in making electronic textiles. In Proceedings of *the International Conference of the Learning Sciences (ICLS '16)*. International Society of the Learning Sciences, Singapore, 498-505.

[8] Renee McCauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: A review of the literature from an educational perspective. *Computer Science Education*, *18*, 2, (June 2008) 67-92.

[9] Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., New York, NY.

[10] Andreas Zeller. 2009. *Why programs fail: a guide to systematic debugging*. Elsevier, Burlington, MA.