Teaching Debugging Collaboratively—Midway Report

Sammy Furr

May 8, 2020

1 Introduction

Debugging is invaluable in writing and understanding code, yet it is rarely formally taught [4]. We typically teach students programming structures, concepts, and languages, but we leave them to learn the tools they use to code by themselves. This approach often works well—the programmer's choice of editor is *very* personal, students figure out how to configure an individualized workflow. Perhaps because debuggers are tools, they often get lumped into the "teach yourself" category. Unlike editors or reference guides however, effectively using a debugger requires a set of high-level, platform agnostic, teachable skills. Teaching these skills is effective, and translates into better, faster, debugging and programming [3] [5].

The use of a debugger is particularly important in a low-level programming classes, which is often the first time that students encounter concepts like assembly instructions and memory addresses. Debuggers such as GCC offer incredibly powerful tools to step through and inspect running code. Sadly, low level debugging tools are often less than intuitive, and provide little to no means for collaboration. Powerful tools such as RR exist that enable collaborative "record and replay" debugging [6], but they lack tools to visualize address spaces and control flow of programs. Research shows that visualization of these previously unexplored spaces aids students taking low-level or systems programming classes [7].

The necessity for accessible, collaborative tools for teaching all aspects of computer science, not the least debugging, has been exemplified by the current COVID-19 crisis. Many students don't have access to a high powered computer (or even a computer at all) to run tools that allow for collaboration and replay by capturing the entire state of a virtual machine [2]. Though the limitations of interacting with a debugger through a phone or low-powered laptop seem daunting, I believe they are actually constraints that can help better shape a visualized, collaborative approach.

For my senior project, I want to develop a front-end interface to RR that enables collaborative debugging and process visualization on a variety of non-traditional platforms.

2 The Value of Teaching Debugging

Before creating a platform for teaching debugging it's of course important to make sure that teaching debugging is actually valuable. There is not a lot research specifically into the efficacy of teaching debugging for computer science students, despite a recent rise in the inclusion of debugging in "computational thinking" curriculums [5]. These curriculums attempt to teach skills in computer science classes that are useful in other subject areas: the UK's computer science curriculum considers debugging an essential "transferable skill" [1]. Though there seems to be confidence that the problem-solving techniques used in debugging are widely applicable, I am more interested in whether systematically teaching debugging actually benefits computer science students. Michaeli and Romeike conducted a good, albeit somewhat small, study on the efficacy of teaching a systematic debugging process to K12 students. They found that students who have been taught a specific debugging framework performed better in debugging tests and were more confident in their own debugging skills [5]. Their result is positive evidence towards the efficacy of teaching debugging, though their research doesn't include college or university students.

As Michaeli and Romeike point out, there is a lack of research into the value of teaching debugging in higher education.

Research into how to best teach debugging is self-admittedly sparse. Chan et al. allow that "in general research on how to improve debugging is sporadic"—an observation that leads them to research a framework to reduce the complexity of teaching debugging [3]. To organize their framework, they split debugging knowledge into 5 categories: Domain, System, Procedural, Strategic, and Experiential. They then review different debugging tools and teaching aids—from those that involve writing code to games—and map tools to the knowledge areas they seek to address. After an evaluation of a host of different tools, they claim a few significant faults in current debugging teaching platforms. The two of which I seek to address follow:

- A lack of tools addressing system knowledge (an understanding of the program to be debugged).
- A lack of back-tracing ability/coverage.

References

[1] Brown, N. C. C., Sentance, S., Crick, T., and Humphreys, S. Restart: The resurgence of computer science in uk schools. *ACM Trans. Comput. Educ. 14*, 2 (June 2014).

[2] Dolan-Gavitt, B., Hodosh, J., Hulin, P., Leek, T., and Whelan, R. Repeatable reverse engineering with panda. In *Proceedings of the 5th Program Protection and Reverse Engineering Workshop* (New York, NY, USA, 2015), PPREW-5, Association for Computing Machinery.

[3] Li, C., Chan, E., Denny, P., Luxton-Reilly, A., and Tempero, E. Towards a framework for teaching debugging. In *Proceedings of the Twenty-First Australasian Computing Education Conference* (New York, NY, USA, 2019), ACE 19, Association for Computing Machinery, p. 7986.

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

[5] Michaeli, T., and Romeike, R. Improving debugging skills in the classroom: The effects of teaching a systematic debugging process. In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education* (New York, NY, USA, 2019), WiPSCE19, Association for Computing Machinery.

[6] O'Callahan, R., Jones, C., Froyd, N., Huey, K., Noll, A., and Partush, N. Engineering record and replay for deployability: Extended technical report. *CoRR abs/1705.05937* (2017).

[7] Walker, J., Wang, M., Carr, S., Mayo, J., and Shene, C.-K. A system for visualizing the process address space in the context of teaching secure coding in c. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2020), SIGCSE 20, Association for Computing Machinery, p. 10331039.