Leveraging Learners for Teaching Programming and Hardware Design at Scale

Leveraging Learners for Teaching Programming and Hardware Design at Scale

Elena L. Glassman
MIT CSAIL
Cambridge, MA 02139, USA
elg@mit.edu

Robert C. Miller
MIT CSAIL
Cambridge, MA 02139, USA
rcm@mit.edu

CSCW '16 Companion, February 27 - March 02, 2016, San Francisco, CA, USA
ACM 978-1-4503-3950-6/16/02.
http://dx.doi.org/10.1145/2818052.2874319

Abstract
In a massive open online course (MOOC), a single programming or digital hardware design exercise may yield thousands of student solutions that vary in many ways, some superficial and some fundamental. Understanding large-scale variation in student solutions is a hard but important problem. For teachers, this variation can be a source of pedagogically valuable examples and expose corner cases not yet covered by autograding. For students, the variation in a large class means that other students may have struggled along a similar solution path, hit the same bugs, and can offer hints based on that earned expertise. We developed three systems to take advantage of the solution variation in large classes, using program analysis and learnersourcing. All three systems have been evaluated using data or live deployments in on-campus or edX courses with thousands of students.

Author Keywords
crowdsourcing; learnersourcing; learning at scale; education

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

Introduction

Learners individually solving programming or digital hardware design problems can collectively generate a wide variety of possible bugs and solutions. We have developed three systems to explore these many bugs and solutions and make the variation useful to teachers and fellow students. All three systems have been evaluated using data or live deployments in on-campus or edX courses with thousands of students.

OverCode [3] visualizes thousands of programming solutions using static and dynamic analysis to cluster similar solutions. It lets teachers quickly develop a high-level view of student understanding and misconceptions and provide feedback that is relevant to many student solutions.

Foobaz [1] clusters variables in student programs by their names and behavior so that teachers can give feedback on variable naming. Rather than requiring the teacher to comment on thousands of students individually, Foobaz generates personalized quizzes that help students evaluate their own names by comparing them with good and bad names from other students.

ClassOverflow [2] collects and organizes solution hints indexed by the autograder test that failed or a performance characteristic like size or speed. It helps students reflect on their debugging or optimization process, generates hints that can help other students with the same problem, and could potentially bootstrap an intelligent tutor tailored to the problem.

OverCode

In MOOCs, a single programming exercise may produce thousands of solutions from learners. Understanding solution variation is important for providing appropriate feedback to students at scale. The wide variation among these solutions can be a source of pedagogically valuable examples, and can be used to refine the autograder for the exercise by exposing corner cases. We developed OverCode to visualize and explore thousands of small Python programs that solve the same problem. OverCode uses both static and dynamic analysis to cluster similar solutions, and lets teachers further filter and cluster solutions based on different criteria. We evaluated OverCode against a non-clustering baseline in a within-subjects study with 24 teaching assistants, and found that the OverCode interface allows teachers to more quickly develop a high-level view of student understanding and misconceptions, and to provide feedback that is relevant to more student solutions.

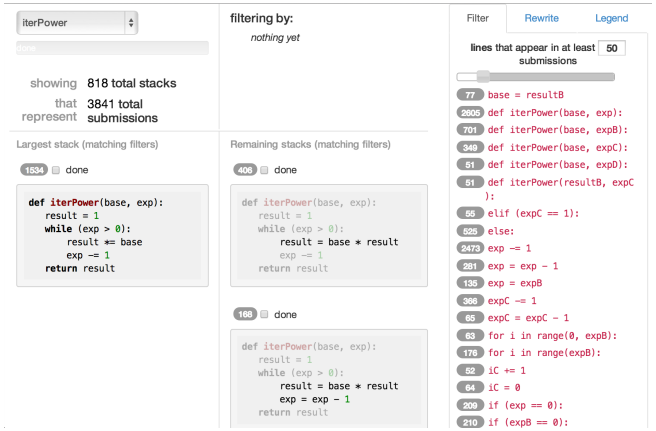

Figure 1: The OverCode user interface. The top left panel shows the number of clusters, called *stacks*, and the total number of solutions visualized. The next panel down in the first column shows the largest stack, while the second column shows the remaining stacks. The third column shows the lines of code occurring in the cleaned solutions of the stacks together with their frequencies.

Foobaz

Traditional feedback methods, such as hand-grading student code for substance and style, are labor intensive and do not scale. We created a new user interface that addresses feedback at scale for a particular and important aspect of code quality: variable names (see Figure 2). Foobaz distinguishes variables by their behavior in the program, allowing teachers to comment not only on poor names, but also on names that mislead the reader about the variable's role. We ran two lab studies of Foobaz, one with teachers and the other with students. In the first study, 10 Python teachers used Foobaz to comment on variable names in thousands of student solutions from an introductory programming MOOC. In the second study, 6 students composed fresh solutions to the same programming problems, and immediately received personalized variable-name quizzes composed in the previous user study.

ClassOverflow

Personalized support for students is a gold standard in education, but it scales poorly with the number of students. Prior work on *learnersourcing* presented an approach for learners to engage in human computation tasks while trying to learn a new skill. Our key insight is that students, through their own experience struggling with a particular problem, can become experts on the particular optimizations they implement or bugs they resolve. The students can then generate hints for fellow students based on their new expertise. ClassOverflow uses new workflows to harvest and organize students' collective knowledge and advice for helping fellow novices through design problems in engineering (see Figure 3). ClassOverflow was evaluated in an undergraduate digital hardware design class with hundreds of students. We show that, given our design choices, students can create helpful hints for their peers that augment or even

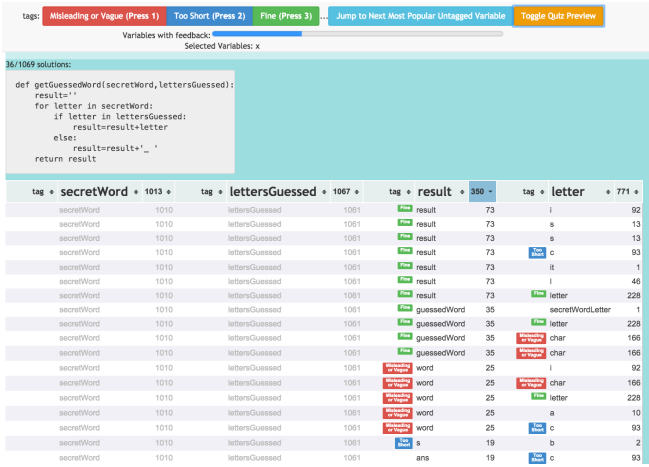

Figure 2: The Foobaz teacher interface. The teacher is presented with a scrollable list of normalized solutions, each followed by a table of student-chosen variable names. Some names shown here have been labeled by the teacher as "misleading or vague," "too short," or "fine."

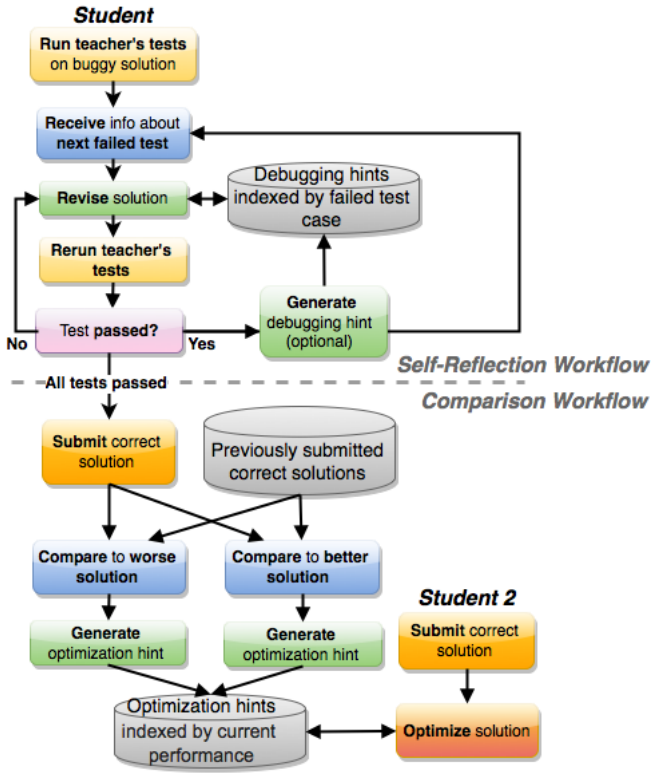

Figure 3: In the *self-reflection* workflow, students generate hints by reflecting on an obstacle they themselves have recently overcome. In the *comparison* workflow, students compare their own solutions to those of other students, generating a hint as a byproduct of explaining how one might get from one solution to the other.

replace teachers' personalized assistance, especially when that assistance is not available.

Discussion

Learnersourcing has been a recurring topic at CSCW recently, and these systems show various mechanisms for leveraging learners in large engineering classes. Learners produce many variations of solutions to a problem, running into common and uncommon bugs along the way. Learners can be part of a closed system workflow that prompts them to generate analysis of their own activity and sends it to selected fellow learners as feedback. Alternatively, learners can be pure producers whose activity is analyzed by systems and distilled by teachers into personalized feedback for fellow learners. We would like to demo these systems together, as a suite of learnersourcing systems that allow teachers to turn the challenges of teaching at scale into an opportunity for discussion, self-reflection, peer-teaching, and more learning from examples.

References

[1] Elena L Glassman, Lyla Fischer, Jeremy Scott, and Robert C Miller. 2015a. Foobaz: Variable Name Feedback for Student Code at Scale. In *Proceedings of the 28th annual ACM symposium on User Interface Software and Technology (UIST '15)*. ACM, New York, NY, USA.

[2] Elena L Glassman, Aaron Lin, Carrie J Cai, and Robert C Miller. 2015b. Learnersourcing Personalized Hints. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '15)*. ACM, New York, NY, USA.

[3] Elena L Glassman, Jeremy Scott, Rishabh Singh, Philip J Guo, and Robert C Miller. 2015c. OverCode: Visualizing variation in student solutions to programming problems at scale. *ACM Transactions on Computer-Human Interaction (TOCHI)* 22, 2 (2015), 7.