Learner-Sourcing in Engineering Classes at Scale

Elena L. Glassman
MIT CSAIL
elg@mit.edu

Chris Terman
MIT CSAIL
cjt@mit.edu

Robert C. Miller
MIT CSAIL
rcm@mit.edu

ABSTRACT
Peer-instruction, peer-reviewing, and more recently learner-sourcing can help students get timely feedback and assistance when the student-to-staff ratio is high. We describe three deployed learner-sourcing scenarios within the context of an undergraduate digital design class. The key design principle is that students should be directed to hints written by other students who have just completed the task themselves. Initial results are promising, and warrant follow-up work.

Author Keywords
engineering education; crowd-sourcing

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION

Professors who repeatedly teach the same large engineering course have the luxury of seeing hundreds of students struggle with, and then complete, the same or similar assignments year after year. Like experienced doctors making their rounds through the computer lab, these professors have seen many of the designs and bugs before, and amaze students with their ability to see the problem, and the solution, right away. The steadily changing pool of teaching assistants have a fraction of that experience. Students are often waiting for help from the central repository of knowledge, the professor, who has limited time and attention, and the teaching assistants, who are more plentiful but less omniscient. And yet the students themselves are each becoming experts on their own bugs and designs.

We are developing systems which harvest and organize students' collective knowledge about engineering assignments. These systems have several potential benefits. Students can experience learning gains by generating explanations [**?**]. While students do not have the pedagogical content knowledge to necessarily generate the optimal explanations, they also do not suffer from the curse of knowledge. Finally, they might be loathe to just give a fellow student the answer, because they themselves worked hard to get it.

L@S'14, March 4–5, 2014, Atlanta, GA.
ACM 978-1-4503-2669-8/14/03
((DOI HERE))

We present three case-studies deployed in an undergraduate computer architecture class called 6.004: Computation Structures. It is taken by over two hundred students each semester, who are typically MIT sophomores and juniors majoring in Electrical Engineer and Computer Science (EECS). Over the course of the semester, students build entire simulated processors composed of logic gate primitives. These designs are expressed as pages' worth of an in-house declarative hardware description language, arranged into their own increasingly sophisticated interconnected submodules. 6.004 is designed such that students first implement digital circuits that work correctly with respect to a battery of public tests. Then students revise their digital circuits to make them better: faster, smaller, or both.

We have deployed systems to harvest and organize students' knowledge for both the hurdles of correctness and optimality. The key design principle is that students should be directed to hints written by other students who have just completed the task themselves. We continue to developing design principles from these experimental, evolving deployments.

RELATED WORK

Peer-Instruction

Summarize ...

Learner-Sourcing

Summarize Juho's work.

LEARNERS EVALUATE ALTERNATIVE DESIGNS

Through exploration of hundreds of previous students' solutions, we found that the space of alternative correct circuit designs is nearly completely separable by the number of device primitives, i.e., transistors, in each design. Picking from previous students' designs, we were able to automatically present current students with design alternatives that were better or worse than their own. Students in the Spring '14 offering of the course were asked to give advice to a future student about how to improve the poorer of the two designs. Their explanations gave a rich window into their own understanding, while serving as strikingly cogent advice to future students.

LEARNERS CURATE DEBUGGING HINTS

Given the complexity of students' simulated processors, students who have recently resolved a bug can be in a better position than a staff member to help a fellow student with the same bug. Bugs are revealed by an incorrect voltage level on a particular wire at a particular time during a staff-provided battery of sequential instructions. While the course already has a vibrant Piazza forum, its generic forum structure was

time-consuming to search for questions, and answers, about debugging a particular verification failure.

Dear Beta is a low-fidelity system we implemented with Google Apps Scripts, built as a central repository of debugging advice for and by students. The processor in this class is called the Beta, and the system's name comes from the fact that it looks like a spreadsheet with an advice column. Students can view other students' hints, which are organized using the standardize wire name and simulated time at which the incorrect voltage level was reported. This makes it easy for students to find appropriate hints within the pages of student and staff-generated hints. Students can post an explanations of their own bugs as they encounter and resolve them, but they must always specify the wire and time at which that bug caused verification to failed, so that the hint is easily found by others.

Providing the explanation is pedagogically useful. Bugs detected by a specific test may have multiple causes, and crowdsourcing may generate hints about all the possible causes. When students sought help and found one of their fellow students' hints helpful, they had the option of upvoting it. System usage statistics, interviews with teaching staff, and anecdotal evidence, including a student's unprompted class forum thank you note, suggest that many students find the system to be a consistently helpful tool.

LEARNERS CROWDSOURCE OPTIMIZATION HINTS

As a final exercise, students optimize their own simulated processors. Students hear stories from classmates of the form, "I tried *x* and got *y* points!" One student heard that two classmates got significantly different outcomes from implementing the same two optimizations in opposite orders. This does not make sense, technically, but these kinds of optimization tales, and a long list of optimization hints written by staff, are all students have had to go on.

Our most recently deployed low-fidelity system is an adaptation of *Dear Beta* for this optimization portion of the course. Rather than indexing hints based on failed test cases, as was done for debugging, we indexed hints based on whether they were intended to reduce a processor's size, minimum clock cycle time, or both. We also gave students the option of submitting the magnitude of the speed and size improvement they got from acting on a hint, so that future students could gauge which optimization hints were most beneficial.

Unfortunately, the effort of acquiring these statistics about the Beta was not trivial, and therefore very few students submitted their data to support this feature. It also could take hours to act on a single optimization hint, so activity on the system was already low. Finally, it did not support hints about failed test cases created during the implementation of an optimization. As a result, this version of *Dear Beta* did not become an instrumental tool for processor optimization.

FUTURE WORK

- deploy 'real' version of Dear Beta system for 6.004x, which does double-duty for optimization as well, since that involves bugs too
- do git diffs on optimization of betas this term to see what folks actually did this term?
- create separate interface in which students can see their own place in space and time, along with diffs, and the ability to comment what they did (via API), with pull-down suggestions from other folks ¡– more of a personal help system that motivates them to annotate.

How best do we close this loop, so that students benefit from the design alternatives and advice generated by classmates?

ACKNOWLEDGMENTS

We'd like to thank the NSF Graduate Research Fellowship and the Amar Bose Teaching Fellowship for funding this work.

REFERENCES FORMAT

References must be the same font size as other body text.

REFERENCES