Learner-Sourcing in an Engineering Class at Scale

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

Christopher J. Terman
MIT CSAIL
32 Vassar St.
Cambridge, MA 02139 USA
cjt@mit.edu

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

L@S 2015, March 14–18, 2015, Vancouver, BC, Canada.
ACM 978-1-4503-3411-2/15/03.
http://dx.doi.org/10.1145/2724660.2728694

Abstract
Teaching computer architecture as a hands-on engineering course to approximately 250 MIT students per semester requires a large, dedicated teaching staff. This Spring, a shortened version of the course will be deployed on edX to a potentially far larger cohort of students, without additional teaching staff. To better support students, we have deployed developmental versions of three learner-sourcing systems to as many as 500 students. These systems harvest and organize students' collective knowledge about debugging and optimizing solutions. We plan to deploy and study the next iteration of these systems on edX this Spring.

Introduction
One-on-one tutoring has been established as a costly gold standard in education [1]. In MOOCs, the teacher-to-student ratio is drastically lower. Teaching staff cannot walk around the lab to get a sense of common success and failures. They also suffer from the "curse of knowledge": the difficulty experts have when trying to see something from a novice's point of view [2].

We are developing systems that harvest and organize students' collective knowledge about debugging and optimizing solutions. Students themselves are each becoming experts on their own bugs and solutions. The

key design principle is that students write hints immediately after completing a task or fixing a bug. Later, other students can use these hints to help guide them to a correct solution or better design. While students do not have the pedagogical content knowledge to necessarily generate optimal explanations, they also do not suffer from the curse of knowledge. These systems relieve some of the pressure on the relatively small teaching staff, and give students the valuable educational experiences of reflection and generating explanations [3].

We present ongoing case studies of three systems deployed in an undergraduate computer architecture course at MIT. Around 250 students enroll each semester. The three deployed systems have distinct objectives: (1) learner-source hints for debugging, (2) learner-source hints for optimization, and (3) prompt students to reflect on other students' solutions. Learner-sourcing refers to crowd-sourcing within the community of students enrolled in a course.

These systems have already been deployed at scale to as many as five hundred students over multiple semesters. Our case studies prepare us to design, deploy, and study the next iteration of these systems on edX this Spring. Throughout the process, we continue to develop design principles that can guide future learner-sourcing systems for engineering classes.

Related Work

In order to generate hints for others, students must reflect on their bug, solution, or optimization, and then explain. We review relevant literature at the intersection of learning theory and engineering education.

Reflection

Reflection and confusion are both treated at length in learning theory literature. Piaget theorized that cognitive disequilibrium, experienced as confusion, could trigger learning: the creation or restructuring of knowledge schema [8]. However, D'Mello et al. point out that, for this learning to take place, it is important for confusion to be both appropriately injected and resolved [5]. Dewey theorized that reflection is a critical method for triggering that transformation from conflict and doubt into clarity and coherence [4]. Turning that reflection into a self-explanation also improves understanding [3]. Turns et al. [11] argue that the absence of reflection in traditional engineering education scholarship is a significant gap. In this work, we aim to design scalable automated opportunities for students to reflect in an engineering course.

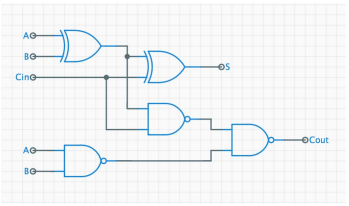
Peer Instruction and Assessment

While reflection is valuable in its own right, it is also a building block of larger frameworks, like Peer Instruction [9] and Peer Assessment [10]. Reflecting on a peer's conceptual development or alternative solution may bring about cognitive conflict that prompts reevaluation of the student's own beliefs and understanding [7]. We aim to trigger productive cognitive conflict that students can attempt to resolve through written reflection.

Generating Hints for Students

With a mixture of automation and human input, helpful hints have been delivered to students in multiple problem domains. HelpMeOut [6] tracks the changes programmers make when fixing a compilation error or runtime exception. Users can write helpful messages to accompany these automatically extracted bug fixes. Our work uses a

gates: 21, fets: 96



gates: 24, fets: 114 (expected solution)

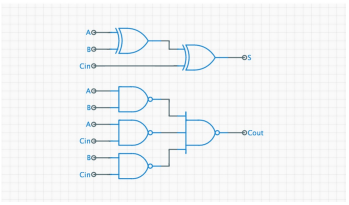


Figure 1: Two correct student solutions (Full Adders), composed of different numbers and arrangements of XOR and NAND gates. The more optimal solution has only 21 gates and 96 transistors (FETs) while the expected, less optimal solution has 24 gates and 114 transistors.

similar strategy for learner-sourcing hints for a new domain. We also learner-source optimization advice.

Debugging

In our course, students design entire simulated processors composed of logic gates. These processors can be challenging to debug even with the one-on-one help of seasoned teaching staff. Students who have recently resolved a particular bug can be in a better position than available staff members to help a fellow student with the same bug. Bugs are revealed by verification failures found by a staff-designed testbed. There are hundreds of possible unique verification failures. The course forum's generic structure does not lend itself to searching for other students' difficulties with particular verification failures.

We built *Dear Beta*, a system that serves as a central repository of debugging advice for and by students, indexed according to verification failures. As soon as a student resolves a bug, they can post an explanation of their bug on Dear Beta along with the verification failure it caused. Students can easily look up advice for their particular failure. Students upvote hints they found helpful. Initial evidence suggests it was consistently helpful. One teaching assistant said, "Whenever I went to help a student, I first asked them if they'd checked Dear Beta." We will create a Meteor or Django-backed version for edX students this Spring.

Optimization

Students optimize their own processors for additional points, which are a function of the processor's size and speed. Students also get a long list of optimization hints written by staff that students consult when choosing which optimizations to implement. Students can take hours to act on a single optimization hint, without knowing which has the greatest payoff.

We deployed a system for learner-sourcing optimization hints and results, with the aim of giving students more transparency and assistance. We indexed hints according to whether they were intended to reduce the processor's size, increase its speed, 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 on average.

This system was not as successful as Dear Beta. We are, however, confident that students are ultimately capable of generating high-quality processor optimization hints, based on results shown in the next section on reflection and comparison.

Reflection

Early on in our course, students design a piece of their processor: a working Full Adder. Students create a variety of solutions that fit the behavior specification. Some are unnecessarily large, and some optimize to use as few transistors as possible. In this scenario, fewer transistors translate to better performance.

Through exploration of hundreds of previous students' solutions, we picked a representative set. Each student was then shown their own solution next to (1) a worse solution in the representative set and (2) the best solution in the representative set. Students were asked to give a hint to future students about how to improve the poorer solution in each pairing (see Figure 1). When the student's own solution is the better solution in the pair, then the student can hint at what the peer had conceptually missed. For example, *Remember DeMorgan's*

Learner-sourced Advice
"Do not try to be too clever with C_{out}—design your schematic as the expression is written. This way you will achieve the [standard] schematic."
"Mutate the boolean function for C_{out} such that all OR and AND operations are being NOT'ed. This allows you to design a circuit using only naturally inverting CMOS gates."
"I would ask: is there a way for you to use some intermediate node in one circuit to bypass a CMOS gate in the other, leading to a reduction of used mosfets?"

Table 1: Examples of learner-sourced advice for optimizing fellow students' Full Adders.

Law: you could replace the 'OR' of 'ANDs' with a 'NAND' of 'NANDs.' When the students' own solution is the poorer solution in the pair, they are challenged to first understand how the better solution uses fewer transistors and then write a hint about the insight for a peer.

This reflection and explanation process is pedagogically valuable on its own. In addition, students' explanations give a rich window into their understanding, while serving as strikingly cogent potential advice to future students (see Table 1). The online version of this course may afford us the opportunity to deliver these optimization hints back to students.

Conclusions and Future Work

This work in progress is the accumulation of several semesters of system development and deployment in a large undergraduate engineering course. Students can write high-quality optimization advice when their solution is paired with a distinct peer solution. Learner-sourced debugging hints, indexed by verification failures, are a valuable student resource. We are preparing to deploy the next iteration of these systems on edX this Spring, while considering how best to measure impact on the student learning experience.

Acknowledgments

We appreciate support from NSF GRF No. 1122374, Quanta Computer, and the Amar Bose Fellowship.

References

[1] Bloom, B. S. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher* (1984), pp. 4–16.

[2] Camerer, C., Loewenstein, G., and Weber, M. The curse of knowledge in economic settings: An experimental analysis. *Journal of Political Economy 97*, 5 (1989), pp. 1232–1254.

[3] Chi, M. T., De Leeuw, N., Chiu, M.-H., and Lavancher, C. Eliciting self-explanations improves understanding. *Cognitive Science 18*, 3 (1994), pp. 439–477.

[4] Dewey, J. *How we think: A restatement of the relation of reflective thinking to the educational process*. D.C. Heath and Company, 1933.

[5] D'Mello, S., Lehman, B., Pekrun, R., and Graesser, A. Confusion can be beneficial for learning. *Learning and Instruction 29* (2014), pp. 153–170.

[6] Hartmann, B., MacDougall, D., Brandt, J., and Klemmer, S. R. What would other programmers do: suggesting solutions to error messages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2010), pp. 1019–1028.

[7] Kavanagh, L., and O'Moore, L. Reflecting on reflection - 10 years, engineering, and UQ. In *Proceedings of the Conf. of the Australasian Assoc. for Engineering Education: To Industry and Beyond*, Institution of Engineers, Australia (2008).

[8] Kibler, J. Cognitive disequilibrium. In *Encyclopedia of Child Behavior and Development*, S. Goldstein and J. Naglieri, Eds. Springer US, 2011, p. 380.

[9] Mazur, E. *Peer Instruction: A User's Manual*. Series in Educational Innovation. Prentice Hall, 1997.

[10] Topping, K. Peer assessment between students in colleges and universities. *Review of Educational Research 68*, 3 (1998), pp. 249–276.

[11] Turns, J., Sattler, B., Yasuhara, K., Borgford-Parnell, J., and Atman, C. Integrating reflection into engineering education. In *Proceedings of the ASEE Annual Conference and Exposition*, ACM (2014).