Open sourcing the classroom

[extended abstract]

Mike Izbicki
UC Riverside
900 University Ave.
Riverside, CA 92521
mike@izbicki.me

ABSTRACT

At UC Riverside, we've created a course on open source software construction called CS100. The course is similar to the standard project-based software construction course, with the exception that projects must be developed and released on GitHub. It is a required course for sophomores. Our main pedagogical contribution is that we open sourced *everything* about the class, including creating the first "open source textbook." The textbook is hosted in a git repository that students are required to contribute to throughout the term. Currently, 88% of the textbook is written by students, including many of the assignments.

We evaluate the course's success in three ways. First, students self report a strong desire to contribute to open source projects and a high confidence in their ability to do so. Second, students have high levels of participation in social networks related to open source software (e.g. GitHub, StackOverflow, Reddit, and Hacker News). Third, student projects have received tens of thousands of downloads from around the world.

1. CS100: COURSE OVERVIEW

The goal of CS100 is to prepare students to develop their own programs. In the course, students undertake their first longterm project: implementing a significant portion of the Unix shell. The course is taken immediately after their data structures course and is a prerequisite for all upper division major courses.

Over the 2014-15 school year, we extended CS100 to include a new goal: to teach students how to effectively use and contribute to open source projects. Open source software is becoming increasingly important in both industry and academia. Large companies like Facebook, Google, and Twitter are depend on open source products to keep their infrastructure running. A great way for students to get hired at the companies is to have a portfolio of open source contributions that demonstrate their knowledge.

CS100 successfully achieved this goal. Figures 1.a and 1.b, show that students self report a high confidence in their ability to contribute to open source software. Figure 2 shows that students are actively contributing to the open source community after the end of the class.

The next two sections briefly describe how the course's programming and written projects effectively taught students the techniques needed for open source software. A full description of the course structure can be found on the course webpage[1].

2. THE PROGRAMMING PROJECT

The programming project is to create a Unix shell called `rshell` that implements a subset of the POSIX standard. This project accomplishes three pedagogical goals.

First, the project exposes students to the operating system's interface. To implement a Unix shell, students must understand how to use system calls like `execvp`, `fork`, and `wait`. Understanding how to *use* these system calls prepares students for their future operating systems class where they discuss how these functions are implemented.

Second, the project exposes students to the need for software engineering principles. The `rshell` project takes the full quarter to complete. To keep students on track, the project is divided into the following four assignments: 1. Students create an executable called `rshell` that reads commands from stdin and executes them. 2. Students implement their own version of the `ls` command. 3. Students add support for piping and i/o redirection to `rshell`. 4. Students add the `cd` command to `rshell` and a signal handler to catch `SIGINT` so that `rshell` does not exit when `^C` is pressed.

In order to simulate the development of a real open source project, we explicitly required that students make their `rshell` code public on GitHub. To turn in an assignment, the students would "push" their work to their GitHub repository; the grader would then "clone" the repo to download the submission. Git is a notoriously difficult tool for beginners to learn, so several labs were dedicated just to git practice before the actual assignment deadlines. Still, many students struggled to correctly submit their code and we had to offer them the opportunity to reshow for reduced credit.

The bulk of the project was completed individually. This ensured students were learning the required operating systems material. But many smaller components were completed in teams. For example, students were required to implement the `rm`, `cp`, and `mv` commands in *another* students repository via a "pull request." This would not have been possible if the projects were not open source.

The open source nature of the project came with a major

[1] `http://github.com/mikeizbicki/ucr-cs100`

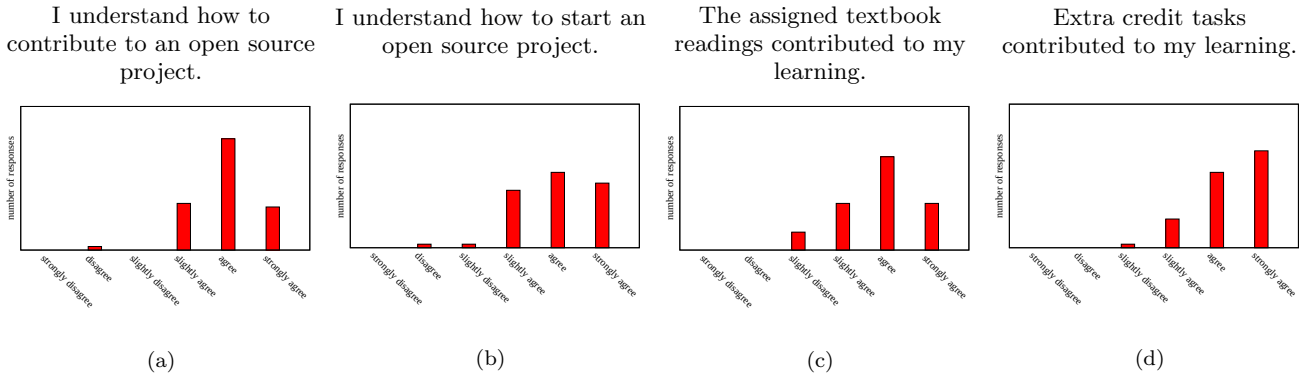

Figure 1: At the end of the winter and spring quarters, we gave students an anonymous survey on the effectiveness of CS100. The results of four questions are shown above. Interestingly, figures (a) and (b) show that students are more comfortable starting their own projects than contributing to another person's project. Many students said they "don't know where to start" when trying to understand a new project. Figures (c) and (d) show that the open source textbook was successful. Students' writings were effective in teaching future students how to accomplish important tasks related to the assignments. The extra credit referred to in figure (d) is fixing mistakes in the textbook.

potential disadvantage: since everyone's code was easily accessible on the internet, plagiarism became much easier. To detect plagiarism, we used the MOSS program[1] combined with git's automatic history tracking features. Overall we found 7 cases of plagiarism out of the 178 students who took the course. This is a comparable rate to other computer science classes.

The last pedagogical purpose of the programming project is to motivate the written project discussed in the next section.

3. THE WRITTEN PROJECT

Students had two options for the written project. The simplest option was to extend the course's textbook. Students were instructed to pick an area of the programming project that they found particularly difficult and write the tutorial "they wished existed when they were working on the assignment." Of the submitted writeups, 3 were incorporated into required lab exercises and 18 become required reading for future students taking CS100. Figure 1.C shows that future students found these readings useful in their learning.

The harder option for students was to create their own open source project and provide documentation for it. Once complete, these projects were advertised on social networking sites like Facebook, Hacker News, and Reddit. Good documentation was the key to making these projects "go viral" and get seen by a large audience. Table 1 lists the projects created by students.

Finally, all students were given the opportunity to earn extra credit by finding and fixing mistakes in the course. Students submitted more than 800 of these corrections. Minor typo fixes were awarded 1 pt of extra credit, but more significant fixes could earn more. Figure 1.d shows that students particularly liked this aspect of the course.

4. REFERENCES

[1] K. W. Bowyer and L. O. Hall. Experience using "MOSS" to detect cheating on programming assignments. In *FIE'99*.

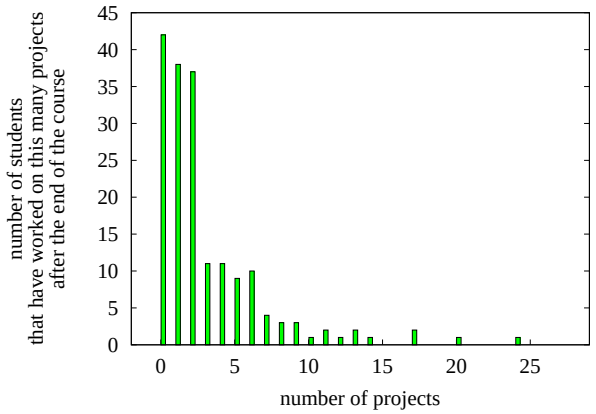

Figure 2: After completing CS100, 136 students (76%) have continued to work on open source GitHub projects outside of class. (GitHub makes user activity publicly available.) Before taking CS100, only 15 students (8%) had ever worked on a GitHub project.

Project Name	Students	Watching	Stars	Forks
BrightTime	2	1	15	2
git-game	2	82	2101	190
git-game-v2	3	17	323	9
manga-tracker	1	1	2	0
Melody-Matcher	3	3	43	2
PacVim	1	26	492	48
regexProgram	2	21	301	50
rhype	1	1	9	0

Table 1: Eight teams of students chose to create and document their own project for the written assignment. The table shows the name of the project (click to visit the project's webpage), number of students who worked on the project, and GitHub statistics about the projects popularity. The projects have cumulatively had more than 30,000 downloads and the projects' documentation has had more than 150,000 visitors from around the world.