FOSS 101: Engaging Introductory Students in the Open Source Movement

Ralph Morelli
ralph.morelli@trincoll.edu

Trishan de Lanerolle
trishan.delanerolle@trincoll.edu

Computer Science Department
Trinity College
Hartford, CT, USA

ABSTRACT

Can engaging students in free and open source software (FOSS) pique their interest in computer science? This paper describes an introductory computer science course that introduced students to using FOSS, to contributing to a humanitarian FOSS project, and to studying the broader impact of FOSS on our society. Students learned basic web programming skills (PHP/MySQL) and made small but significant contributions to a global FOSS project. Mistakes were made and opportunities were missed. But overall the experiment was a success and the experience was enjoyable and educational for students and instructor alike. By building on what worked well, this course could serve as a model for incorporating study of FOSS into the introductory computing curriculum.

Categories and Subject Descriptors

K.3.2 [**Computing Milieux**]: Computer and Information Science Education—*Computer science education, Curriculum*

General Terms

Human factors

Keywords

Open source software, open source movement, curriculum development

1. INTRODUCTION

Free and open source software (FOSS) is having a growing impact on the computing industry, and, more broadly, on modern society. According to a recent study by the Standish group, open source software is "raising havoc throughout the software market" and is estimated to cost the software industry $60 billion in annual revenue[11].

SIGCSE'09, March 3–7, 2009, Chattanooga, Tennessee, USA.

The Mozilla project represents a case in point. As of March 2008, its Firefox browser had captured a 17.83% share of the browser market (see mozilla.org/About). Most of its success has come at the expense of Microsoft's Internet Explorer, which in 2002, when Mozilla 1.0 was released, commanded 95.4% of the browser market[16].

The Firefox project is an excellent example of the FOSS community and development process. Of its 150 million users, between 1000 and 2000 developers have contributed code to the project, more than 10,000 users have served as testers, and 80,000 people are actively involved in recruiting users and contributors (mozilla.org/About). And Mozilla is just one of hundreds of successful large-scale FOSS projects. Currently, the Sourceforge FOSS repository (sourceforge.net) has more than 1.7 million registered members and over 170 thousand registered projects.

But the impact of the FOSS movement doesn't end with software. The success of software development projects like Linux and Mozilla has inspired similar projects in all sorts of domains. The FOSS production model–distributed, peer-based, open, transparent, communal and socially conscious–is being employed in politics, publishing, science, and just about everywhere the Internet reaches[9].

Yochai Benkler has coined the term *commons based peer production* (CBPP) to characterize the FOSS development model as:

> collaboration among large groups of individuals...who cooperate effectively to produce information, knowledge, or cultural goods without relying on either market pricing or managerial hierarchies to coordinate their common enterprise [1].

One of the best examples of CBPP is Wikipedia (wikipedia.org). Started in 2001 by Jimmy Wales and initially greeted by skepticism from academics and librarians because of its democratic model and reliance on amateurs, Wikipedia has grown into an increasingly respected and reliable source of information [15]. Other examples include Architecture for Humanity (architectureforhumanity.org), an organization through which architects contribute their designs freely and openly to "communities in need," and CAMBIA (www.cambia.org), whose Biological Open Source (BiOS) initiative develops and shares life-sciences technologies, such as seeds, that address "clear and real public needs."

Today's undergraduates have grown up with the internet and the web, belong to virtual communities like facebook, share their music, their photos, their video clips, their friends. In general, they are deeply immersed in the open

and sharing culture that the Web has become, without necessarily seeing the connections to the free software movement that, in many ways, initiated that culture. For example, they know about and contribute to Wikipedia, without necessarily knowing that its software platform, MediaWiki (mediawiki.org), is FOSS and its community-based production model is inspired by the FOSS development model. Today's students are also very engaged in community service activities, and therefore share some of the same sentiments towards the public good that one finds in the cooperative and sharing culture of the FOSS movement.

Yet, despite the growing impact of the FOSS development model, and the broader open source movement, FOSS has received relatively little attention in the undergraduate computing curriculum [13]. The question then is whether we can use students' immersion in "things free and open" to interest them in FOSS and some of the computational ideas that underlie this broad cultural movement.

To answer this question, we developed a new course *FOSS 101*, whose primary goal was to introduce students to FOSS and tie its principles and methods to some of the products (Wikipedia) and activities (file sharing, blogging) that characterize contemporary student culture. Our hope was that, once students learned about the connections between FOSS and, say, Wikipedia, they would have a more complete and positive view of computer science as a discipline. Moreover, once these students were introduced to FOSS programming, at least some would find it enjoyable and rewarding and want to take additional CS courses.

2. CONTEXT AND RELATED WORK

FOSS 101, officially known at Trinity College as *CPSC-110-06: Open source software for humanity*, was offered at the introductory level with no prerequisites. It is part of the Humanitarian FOSS (HFOSS) Initiative, an NSF/CPATH-funded effort that is exploring whether engaging students in FOSS development can help revitalize undergraduate computing education[5]. Under the HFOSS umbrella, various approaches have been attempted, including software development courses, independent study courses and summer internships [6, 12, 7]. But, this was the first attempt to introduce the FOSS model at the introductory level. The authors are unaware of other attempts to introduce FOSS at that level.

3. COURSE OVERVIEW

FOSS 101 satisfies a "numeric or symbolic reasoning" distribution requirement and therefore includes a substantial amount of programming. As the following course description suggests, the course had two main goals: (1) introducing students to programming and software development within a FOSS framework, and (2) engaging students in reading and discussion of the broader free and open source movement and its current and potential impact on society.

> **Course Description.** Free and open source software (FOSS) is software that can be modified, customized, and redistributed by users and programmers. From its modest beginnings in the 1970s through the rapid growth of the Internet and the GNU/Linux operating system, today's FOSS movement is a global enterprise involving millions of programmers working together on thousands of software programs. A growing number of FOSS programs have a humanitarian focus. In this course we will both learn about and contribute to a real open source project. We will work with the Sahana[1] system, a crisis management system that was built in the aftermath of the Asian Tsunami. We will learn to write web-based application software using FOSS tools. Come and join the open source movement.

Missed Opportunity: #1: FOSS and Wikipedia In hindsight and based on the experience gained in having now taught the course and seen what caught the students' interests, it is clear that the course description could be made far more effective as a recruiting tool by emphasizing the connections between FOSS and, say, Wikipedia or other familiar "free and open" activities.

3.1 Course Details

The class met twice a week for 75 minutes in a video conference venue and was open to students at Trinity College, Wesleyan University, and Connecticut College.[2] Enrollment consisted of 13 students from two campuses, including four seniors, one junior, five sophomores, and three freshmen. Of these, seven students (two seniors, two sophomores, and three freshmen) had never taken a computer science course before. Two of the students were senior computer science majors. Although the vast majority of students taking the course indicated "interest in the topic" as their primary reason for taking the course, one of the seniors took it to satisfy the distribution requirement.

Mistake #1: Class Schedule Scheduling the course as two 75 minute class sessions left too little time for programming topics.

It would have been more effective to schedule this in a three-day 50-minute time slot. That would allow one class per week for lecturing on programming and technical topics, a second for hands-on lab sessions to work on programming and software development exercises. The third meeting could then be devoted entirely to lecture and discussion of broader social issues.

Mistake #2: Enrollment Diversity The huge diversity among students' backgrounds and interests made it especially challenging to develop appropriate programming instruction and assignments. Several students had no prior programming experience, while several others came with fairly extensive experience in Java or PHP/MySQL.

This problem could be addressed by limiting enrollment to students who have not already taken a computer science course. This would have allowed the course to focus on teaching introductory programming skills without worrying about boring the advanced students.

On the other hand, at least one of the seniors, a female, non-CS student, seemed to enjoy the course:

> I have to say from a non computer science major that I have really enjoyed the class. Never have I talked about a class so much outside of the

[1]Sahana is FOSS disaster management system developed in Sri Lanka following the 2004 Asian tsunami (sahana.lk)

[2]The video conference facilities and three-college collaboration was originally made possible by a Mellon Foundation grant that eventually led to a more focused three-college collaboration in the NSF-funded Humanitarian-FOSS project.

> classroom, I have found it informative and have really learned a lot.

This particular student had no prior programming experience but discovered that she enjoyed programming. One wonders what might have happened had she taken such a course in her first or second year.

3.2 Sharing and Collaboration

To mitigate some of the differences in background, and to mimic the culture and ethic one finds in the FOSS world, an effort was made to encourage an atmosphere of sharing and collaboration, which was expressed thusly in the syllabus:

> Expectations will differ somewhat depending upon what you bring to the course... Students will be expected to contribute at their appropriate skill levels [and] ... those who already know how to program will be asked to help others learn how to program.

Moreover, except for two graded "development projects" on which they worked in pairs or teams, programming homework and exercises were not graded. Students were encouraged to share their solutions, which were posted on the course's Wiki. In class, students were asked to walk us through their solutions and their failures or problems. Significant class time was spent working cooperatively to track down bugs or explore other ways–mostly syntactical variations–of solving a problem.

3.3 Peer Grading?

Grades in the course were determined in the traditional way. There were two examinations (worth 40% of the grade), a term paper (20%), frequent homework assignments and quizzes (20%), and two programming assignments (20%).

Missed Opportunity #2: Peer Evaluation Partly for technical reasons–lack of a suitable support system–we stopped short of adopting a truly peer-based approach to grading, such as described in [3, 8].

Given the extensive reliance on peer assessment in the FOSS development model, using some form of peer evaluation in a course about FOSS would seem to be a natural choice. Peer assessment in computer science and other disciplines has been much studied, although the problems associated with doing it efficiently and well remain challenging [10]. However, in the future it seems clear that some kind of *partial peer evaluation* model would be appropriate for this course, where students evaluate each other in confidence and in conjunction with the instructor's own evaluation. Getting students assessment of their own and their teammates' performance on group projects would seem to be particularly appropriate for a FOSS course.

3.4 Course Topics

Course content was divided between technical topics related to the development of free and open source software (FOSS), including programming in PHP and MySQL, and readings and assignments related to broader issues about the societal impact of the open source movement (Table 1).

Overall, this content was appropriate: the students learned to read and write and understand PHP/MySQL and were generally enthusiastic and engaged in the readings and discussion topics and assignments. The combination of the two allowed for many opportunities to make connections between FOSS and the broader impact of FOSS culture.

Table 1: Course topics.

Technical	Societal
Web-based applications	Open source everywhere
Installing PHP	Wikipedia
PHP Decisions & loops	Citizen journalism
PHP Functions	Open source spying
HTML forms processing	Open source licensing
MySQL & PHP	Commons based peer production
Sourceforge & SVN	Open source biology
Sahana & VMOSS	Humanitarian FOSS

3.5 Homework Assignments

Weekly homework assignments typically consisted of two parts. The first part was a technical reading and programming assignment. The readings came mostly from the textbook [4].

The programming assignments consisted of a sequence of increasingly complex script writing exercises, beginning with PHP scripts using variables and print statements and progressing to scripts that introduced other features of the language (operations, if/else, loops, functions, and arrays). We did not cover object-oriented concepts. The last several assignments introduced HTML form processing, including the use of a MySQL database. Students were encouraged to work in pairs and to share their knowledge and their code with each other.

All students used their own laptops for software development. This required them to download and install a collection of open source development tools, which occupied the first couple of "technical" assignments. Class time was spent working cooperatively to solve various system problems that arose. Often the solutions were discovered by the students themselves, not by the instructor or TA. The development environment consisted of the open source Eclipse IDE (eclipse.org) and a XAMPP (Apache, PHP, and MySQL) stack (xampp.org), installed on either a Windows or Macintosh platform.

The second part of the weekly homework assignments consisted of reading and responding to articles about some aspect of the open source movement. Written responses were posted to the class Wiki and discussed in class. For example, one assignment asked students to read and comment on Rosenzweig's article "Can history be open source?," which addresses the quality of the history and biography entries on Wikipedia [15]. The essays had to be written in *Wikipedia style*, with hyperlinks to important related concepts, and footnotes and references to support their claims.

Another assignment expanded on the peer-production approach by asking students to work collaboratively to produce a Wiki page that provided a comparison of two online news sites, a mainstream site (such as CBS or MSNBC) and a *citizen news* site (such as Wikinews). To simulate the dynamic creation and editing of a Wiki page they were required to work in groups of 3-4 where the groups were created on a first-come, first-served basis. The groups' topics had to be distinct. Anyone could start a group or join another group, but no group could have more than four contributors.

Missed Opportunity #3: Peer Evaluation This would

have been an excellent assignment to experiment with some form of peer evaluation.

3.6 Major Assignments

In addition to weekly homework assignments, there were three major assignments: a 10-12 page term paper, a PHP-MySQL programming assignment, and a FOSS assignment. For the term paper, students could write on any topic related to the open source movement. The topics ranged widely in scope, including open source politics and government, open source science, several papers on the Mozilla project, open source licensing, copyright vs. copyleft. Overall, the papers were quite informative and of good quality. During the last week of the course students gave oral presentations of their papers.

The programming project consisted of developing a simple web application with a MySQL database for inputting and displaying a list of charitable organizations. Students were allowed to work in pairs. None of the pairs involved two novices working on their own, the hope being that novices would learn from those with some prior programming experience. In this case the expectation was that each pair would work out its own solution, with the help of the TA and instructor when questions arose. This assignment was graded in a traditional way.

3.7 The HFOSS Assignment

For the FOSS assignment students were required to make a contribution to VMOSS (Volunteer Management Open Source Software), a Sourceforge-hosted FOSS project, which is a stand alone derivative of the Sahana volunteer management module (sourceforge.net/projects/vmoss/). Because VMOSS was created as part of the HFOSS initiative, working with it gave students an introduction to Sourceforge's project management features without the sort of apprehension that would occur if students tried to join some other Sourceforge-hosted project. This was important–attempting to participate in an open source project can be intimidating even for experienced software developers.

Moreover, the decision to focus on a *humanitarian* FOSS application was a deliberate one. Our hope for the course and more broadly for the HFOSS initiative is to capitalize on students' interest in community service. Indeed, at the beginning of the course, some students were encouraged to take the course as a way of preparing themselves for the HFOSS summer internship program (hfoss.org). Several of the students did apply and one was accepted into the summer program.

In phase one of this project, students were required to download and use the software and report any apparent bugs and/or feature requests to the project's bug-tracking facility on Sourceforge. These were reviewed and discussed in class.

For phase two, students were required to make one of several types of contributions to the VMOSS project, ranging from developing a VMOSS user guide (5 students), to developing a database initialization script in SQL (3), to developing a software patch (1), to conducting a comparative study of similar volunteer management systems (1), to updating a directory of humanitarian FOSS projects (3). The wide range of options was designed to allow for the large diversity among student skills and interests and to illustrate that there are valuable ways to make a contribution to an open source development project that don't require programming.

Table 2: Student Course Evaluations.

Question	Very	Moderately	Not Very
Challenging?	1	6	1
Class worthwhile?	6	1	1
Assignments helpful?	6	2	0

Mistake #3: Ambitious Expectations Some aspects of the FOSS project were perhaps too ambitious for this particular mix of students. The fact that only one student–a female CS student–attempted to create a patch, despite efforts to identify problems that could be solved fairly simply, is an indication that this particular assignment needs more careful design.

The main problem was that VMOSS employs an object-oriented architecture and there wasn't sufficient time in the course to study object-oriented PHP. One possibility would be to introduce the project early in the semester and design homeworks and other exercises more carefully so they tie in directly to the FOSS assignment.

Nevertheless, the assignment as a whole was successful–the students made real contributions to VMOSS and seemed to gain satisfaction from their contributions. The decision to allow several alternative ways of contributing to the VMOSS project proved to be an appropriate way to elicit real contributions from students. To some degree it also helped underscore the FOSS principle–articulated by Eric Raymond as the "buzzing bazaar"[14]–that the FOSS model provides freedom to contribute to projects in ways that best suit one's own interests and skills.

Thus, although there is ample room for improvement here, this type of assignment–getting students to contribute to a real FOSS project–is definitely appropriate for this course.

4. STUDENT COURSE EVALUATIONS

Our perceptions that the course was successful were supported by student course evaluations. Two surveys were used to elicit student feedback. The first was a generic (institutional) course evaluation survey. Eight out of 13 enrolled students returned that evaluation. Seven of the eight reported that "interest in the course material" was one of the reasons they were taking the course. Only one of students listed "distribution requirement" as the only reason for taking the course. Table 2 shows that students were mostly positive about the course's content and intellectual merit.

The second survey, completed by 4 of the 13 students, used open ended questions to elicit student feedback on how the course shaped their perceptions of computer science, FOSS, and the computing profession. A representative sample of their responses suggests a similarly positive response:

- Wow. I really got to look at how computer science can relate to humanitarian efforts. I really loved delving into a world I'd barely seen before: Open Source. I now really understand it and know why it came about.
- Before I thought that [FOSS] was only relevant to computer scientists and no one else. But that is not true. The open source movement is much more relevant to the greater humanitarian good.
- I learned that [Open-source software development] is like helping each other to build software.

- It shows how my input to the open source community can have a greater impact on the open source community with the available source on the web.

- Open-source software development allows for low- or no-cost highly customizable software products that can be used to support many causes who have limited financial means.

- Being involved with this course mainly made me think about how to teach others about HFOSS, but in doing so, I realized the importance of building a community for open source projects.

- I now have a better understanding of what it is like to work with an contribute to a team of people, even when I may never meet them in person.

5. CONCLUSIONS

It would be premature to draw any strong conclusions from such a small-scale experiment. However, overall, our experience does seem to indicate that this type of course could serve as an effective way to attract students, who might not otherwise have shown interest in computer science, to the computing discipline. We conclude with several observations that support this view.

- Students in the course were comfortable with virtual communities and with "free and open" products (Wikipedia) and experiences (blogging, file sharing). Their interest in such activities can serve as hooks into computing topics, such as FOSS.

- Students were not generally aware of the relationship between free and open products and experiences and free and open source software. Educating them about this relationship does appear to mitigate some of their biases and misconceptions about computer science. For one thing, it shows them that building software is a highly social and cooperative activity.

- Students were remarkably patient and accepting of the several "technical difficulties" that arose when installing downloaded software and using new tools for the first time. As experienced Internet users, it may be that they have grown used to such problems.

- Engaging students in a real FOSS project is a valuable way to make the connection between programming and what it accomplishes. It might even serve to teach them that with some additional CS education they could make real contributions to software development projects that they care about.

- Focusing on *humanitarian* FOSS, such as Sahana and VMOSS, is an effective way to tap into our students' contemporary interest in public service and community learning. Assuming the truth of the claims that women are more interested than men in applications that benefit society[2], it may eventually help attract more women to computing courses.

6. ACKNOWLEDGMENTS

This Humanitarian FOSS Project is supported in part by the National Science Foundation under grants #0722137, #0722134, and #0722199. This course also benefited from the use of facilities supported by the Mellon Foundation.

7. REFERENCES

[1] Y. Benkler and H. Nissenbaum. Commons-base peer production and virtue. *Journal of Political Philosophy*, 14(4):394–419, November 2006.

[2] M. Bombardieri. In computer science a growing gender gap. *Boston Globe*, August 2006.

[3] J. D. Chase and E. G. Okie. Combining cooperative learning and peer instruction in introductory computer science. In *Proceedings 32rd SIGCSE Technical Symposium*, pages 139–143, March 2000.

[4] M. Davis and J. Phillips. *Learning PHP & MySQL, Second Edition*. O'Reilly, 2008.

[5] T. de Lanerolle, R. Morelli, N. Danner, D. Krizanc, G. Parker, and O. Izmirli. Creating an academic community to build humanitarian foss: A progress report. In *Proceedings of the 5th International ISCRAM Conference*, pages 337–341, May 2008.

[6] H. J. Ellis, R. Morelli, J. Damon, and J. Raye. Can humanitarian open-source software development draw new students to cs? In *Proceedings 38th SIGCSE Technical Symposium*, pages 551–555, March 2007.

[7] H. J. Ellis, R. Morelli, and G. Hislop. Holistic software engineering education based on an open source humanitarian project. In *Proceedings of the 20th Conference on Software Engineering Education and Training*, pages 327–335, July 2007.

[8] E. F. Gehringer. Electronic peer review and peer grading in computer-science courses. In *Proceedings 33rd SIGCSE Technical Symposium*, pages 139–143, March 2001.

[9] T. Goertz. Open source everywhere. *Wired*, 11(11), November 2003.

[10] E. Z. Liu, S. S. Lin, and S. M. Yuan. Alternatives to instructor assessment. *International Journal of Instructional Media*, 29(4):395–404, 2002.

[11] Marketwire. Free open source software is costing vendors $60 billion. http://www.marketwire/mw/release.do?id=844462, April 2008.

[12] R. Morelli, H. J. Ellis, T. de Lanerolle, J. Damon, and C. Walti. Can student written software help sustain h-foss? In *Proceedings of the 4th International ISCRAM Conference*, pages 41–44, May 2007.

[13] D. Patterson. President's letter. *CACM*, 49(3):27–30, March 2006.

[14] E. S. Raymond. *The Cathedral and the Bazaar*. O'Reilly, 2001.

[15] R. Rosenzweig. Can history be open source? wikipedia and the future of the past. *Journal of American History*, 93(1):117–146, June 2006.

[16] T. Wang, L. Wu, and A. Lin. The revival of mozilla in the browser war against internet explorer. *Proceedings of the 7th International Conference on Electronic Commerce*, page 159, 2005.