10-20-2011

Free and open source software development of IT systems

Mihaela C. Sabin
University of New Hampshire, Manchester, mihaela.sabin@unh.edu

Recommended Citation
Mihaela Sabin, Free and open source software development of it systems, Proceedings of the 2011 Conference on Information Technology Education (New York, NY, USA), SIGITE '11, ACM, 2011, pp. 27–32.

Free and Open Source Software Development of IT Systems

Mihaela Sabin
Computer Information Systems
University of New Hampshire
Manchester, NH 03101
603 641 4144
mihaela.sabin@unh.edu

ABSTRACT

IT system development, integration, deployment, and administration benefit significantly from free and open source software (FOSS) tools and services. Affordability has been a compelling reason for adopting FOSS in computing curricula and equipping computing labs with support infrastructure. Using FOSS systems and services, however, is just the first step in taking advantage of how FOSS development principles and practices can impact student learning in IT degree programs. Above all, FOSS development of IT systems requires changes to how students, instructors, and other contributors work collaboratively and openly and get involved and invested in project activities.

In this paper I examine the challenges to engage students in FOSS development projects proposed by real clients. A six-week course project revealed problems with adopting FOSS development and collaboration across different activities and roles that student team members have assumed. Despite these problems, students have showed a genuine and strong interest in gaining more practice with FOSS development. FOSS development teaching was further refined in two other courses to learn about adequate teaching strategies and the competencies that students achieve when they participate in FOSS development of IT systems.

Categories and Subject Descriptors

K.3.2 [**Computer Information Science Education**]: Computer science education, Curriculum.

General Terms

Human Factors, Experimentation, Management, Performance.

Keywords

Free and open source software, IT system development, collaboration.

1. BACKGROUND

The Computer Information Systems (CIS) program at University of New Hampshire in Manchester (UNHM) has built partnerships with local nonprofits, state agencies, small businesses, and in-house research projects to enrich curricula with authentic work experiences and benefit community with IT solutions. In this curricular model, 98 students worked in 26 teams on projects proposed by eight organizations and one research faculty during 2007 to 2010 time period. Most of the projects evolved over multiple semesters as course projects in databases, web systems, and software engineering courses. Three projects were conducted in internship, independent study, and summer research courses by individual students. Only these projects have produced prototypes that could be transferred for deployment at partnering organizations. Two of these three projects have been adopted and are in use at sponsoring organizations, Salvation Army in Manchester and the Division for Juvenile Justice Services.

Integrating real-world projects in the IT curricula has widely-recognized advantages on student learning:

1. *Relevant learning*. Students solve practical IT problems that are encountered and formulated by community partners.
2. *Deeper learning*. Students demonstrate higher order cognitive competencies, such as synthesis and evaluation.
3. *Collaborative learning*. Students work in teams. Collaboration and communication are integral part of their work.

However, the implementation of this model has revealed some critical weaknesses:

1. Low rate of success with building useful prototypes.
2. Ineffective collaboration and communication, especially in coding activities.
3. Less attention to the actual artifacts and more interest in describing individual activities in the students' self-evaluations, whether reported development activities were successful or not.
4. Very limited reusability of the system's artifacts (models, documents, source code, user support manuals) between project iterations across semesters.

These weaknesses stem from a visible polarization between, on one hand, requirements elicitation, analysis, and design activities and, on the other hand, implementation and deployment activities. Documents and models that students produced were not very effective in guiding the writing of source code and planning and conducting testing experiments, despite efforts made to apply iterative and agile software engineering methods. Students themselves felt that one either loves programming or hates it, and participation in non-programming related activities was perceived as orthogonal to implementation activities. This phenomenon has even a more significant negative impact when artifacts that were produced in a certain course have to be reused in following

semester and different course to make progress with system development in a subsequent iteration.

The research question these shortcomings raise is *how to actually apply FOSS development principles and practices and change student attitudes towards a much more responsible collaboration to engage them productively and successfully in FOSS development of IT systems*.

My personal experience with FOSS [1] and Humanitarian FOSS [2, 3] convinced me that the FOSS culture, principles, and practices are very much suitable for a student-centered educational environment that is inquiry-based, highly collaborative, motivational and relevant, and inclusive of diverse abilities, cultural backgrounds, and life experiences.

In the fall 2010 semester I decided to rethink the team projects in an introductory course in web development and adopt a FOSS development approach with two goals in mind:

1. To improve implementation activities and integrate them more productively with the other software development activities.
2. To increase student interest in and responsibility for challenging system development and project management activities.

The rest of the paper describes the plan for incorporating FOSS activities in a fall 2010 course, reports on how these activities scaled up in two other courses in spring 2011, and outlines future work that has been informed by the lessons I learned.

2. PLANNING STEPS

Starting in fall 2010, the CIS 505 Advanced Web Authoring course has become a required course in the CIS major. The course enrolls sophomores in the program and juniors who have transferred with an associate's degree from local community colleges. The course prerequisite is an Intro to Internet and Web Authoring course that fulfills the general education requirement in the Environment, Technology, and Society curricular group.

2.1 Student Demographics

UNHM is a commuter college. The large majority of students enroll full-time while holding jobs that demand close to 20 hours a week of their time. Two thirds of UNHM students qualify for financial aid. The CIS program has articulation agreements with local community colleges. In 2010-2011, close to half of the students in the program transferred at least 50 credits from two-year colleges. Women representation in the program is 11%, lower than the reported 18% of 2008 computer and information sciences undergraduate recipients who were female [8].

New Hampshire is the fourth whitest state, with minorities accounting for about 7.2% of the state's population in 2009. The most diverse part of the state is the Manchester-Nashua metropolitan area. In 2007, 50% of all the minority residents resided in this area, with 11% of the metro area's population being minority [9].

At the CIS 505 course level, the demographic data from the 20 students who took the class in fall 2010 showed slight variations from the larger picture's data. The class had three females (15%) and two minority students (10%). Other demographic data of interest indicated that there were seven transfer students (35%), four veterans (20%), and two non-CIS majors (20%), a male Business major and a female Communication Arts major with a minor in CIS. The breakdown by year of study had two sophomores, nine juniors, and nine seniors.

The demographics picture informs about the challenges that students have when a course is intentional about adopting a FOSS development approach. Students are expected to spend 6 to 8 hours outside class in activities through which online peer communication is pervasive and collaboration in pairs or within teams of 3 to 6 students becomes extensive at various levels: code base, documentation, and project management. Student time constraints, diversity of academic backgrounds, and the mix of other cultural and personal experiences could be barriers to how communication and collaboration channels work.

2.2 Learning Outcomes and Assessment

In the CIS 505 course students learn to:

- Apply dynamic web programming concepts and techniques
- Create and experiment with web applications and systems
- Review, document, share, test, and deploy web applications
- Use open source collaborative software and content management tools to develop web applications
- Communicate timely and work in teams effectively
- Argue for the use of open source software tools and adoption of open source collaboration practices.

The course learning outcomes are assessed by measuring student performance in two exams (40%), eight weekly homework assignments on which students work in pairs (24%), a team project with four iterations scheduled at the end of the semester (24%), and in- and outside class participation (12%). Students and instructor are the evaluators of student performance. The instructor grades the exams, student participation, and project status reports, which represents 66% of the final grade. Students self-evaluate their homework assignments and submission of peer reviews (28%) and have their contribution to the last two iterations of the project evaluated by peers on other teams (6%). Student grading weighs the remaining 34% of the final grade. Assessors use rubrics for their evaluations.

2.3 Teaching Resources

The course materials were created by examining lecture notes, assignments, exam problems, and textbook suggestions from two colleagues teaching a similar course at other schools. Students used Safari Books Online service offered by the university to access the course textbooks [4, 5]. Two other references were used for the project portion of the course: Fogel's book on producing open source software [6] and RMH Homebase, a volunteer management and scheduling system designed at Bowdoin College for the Ronald McDonald House in Portland, Maine [7].

The course infrastructure included the course site, two wikis, a class forum and mailing list, student online portfolios, a staging server for testing and demo purposes, and central repositories for the projects' code base. WordPress, Google Groups, Google Sites, MediaWiki, BitNami WAMP bundle, SourceForge hosting, and two UNHM CIS servers were used to support the course infrastructure. Live USBs with Apache, MySQL, and PHP package were used in class for lab activities. Software developed on student personal computers or in the lab used XAMPP, NetBeans, and TortoiseSVN.

All course materials were openly available from the course site that I created using WordPress publishing tools and have hosted on the WordPress servers. The course forum and mailing list were set up with a Google group. Students were asked to create online portfolios with Google Sites to assemble their work in the course, write self-evaluations, receive feedback and grades from me, and write their peer reviews. Two MediaWiki wikis were installed and configured to (1) document each team progress with their projects and (2) make available wiki articles on tools and services used in the course.

3. FIRST EXPERIENCE

3.1 Course Projects and Community Partners

The last six weeks of the semester were allocated to software development of web applications that solve IT problems raised by four partners: two nonprofits, YWCA of Manchester and Big Brothers and Big Sisters of Greater Manchester (BBBS), one government agency, the Domestic Violence Unit of the Manchester Police Department (DV Unit), and the UNHM Internship and Community Outreach program.

Four student teams (of four to six students per team) were formed to work on the projects:

- DONATE, to track donors and analyze donations for YWCA and BBBS.
- DVUnitCMS, a counseling management system that prosecutors and counselors in the DV Unit would use to assign and observe progress with their counseling services.
- GoInterns and Internship2Career, same project with two names and two teams, to track internship opportunities.

3.2 Development Roles and Activities

Development responsibilities fell in two categories. The primary role category included responsibilities for requirements and analysis, subsystem design, and subsystem implementation. The supportive role category included the roles of project leader, wiki editor, technical writer, reviewer, and IT support manager. I named the project leaders based on my knowledge of student performance in the course and overall academic standing. I also designated each team's IT support manager. These roles assumed responsibilities for configuring the SourceForge projects and building the team's application on the staging server for peer review and demo presentations. All students were asked to review work products completed by two peers from other teams for the last two iterations of the project. The other student roles were negotiated within the team.

In the end, each team had an analyst, one or two designers of either the user interface or the storage subsystems, and two or three implementers of the application logic or control subsystem. On the supportive roles front, each team had a project leader, a wiki editor, an IT support manager, and one or two technical writers. All team members assumed reviewing responsibilities.

3.3 Results

Two of the four projects, DV Unit CMS and GoInterns, reached the phase where the user interface, persistent storage, and control subsystems were integrated. The teams of these projects had more experienced implementers and closer and more effective collaboration between the PHP and MySQL coders. In the other two teams, DONATE and Internship2Career, effective collaboration took place within the boundaries of individual subsystems, and there was minimal communication among implementers of different subsystems.

With a couple of exceptions, analysts and designers did not want to and were not involved in coding activities. They either felt that they did not have the skills or that the team was lacking the chemistry for truly collaborative work. Their work products were user and installation manuals and developer's guides that outlined the overall development process.

Everybody was expected to include their work products in the team project, whether models, documents, or source code files, and use version control to manage project progress and changes. However, the use of version control for the purpose of keeping everybody on the team informed about individual contributions was, to say the least, problematic. The same phenomenon of minimum use of tools and services that promote sharing was observed about reviewers. Most of them did not check out source code they were supposed to examine from the SourceForge project repositories. Similarly, IT support managers were expected to assist team members or other peers with testing prototypes on the staging server or replicating testing environments on the student computers. Again, there was minimum interaction and investment in an open collaboration to get this kind of work done.

Despite these shortcomings, twelve students have enrolled in a more advanced, project-based, and programming-intensive elective course in spring 2011. This is about twice the number of students who have taken similar courses in the past. It looked like the students and I have just started to scratch the surface of FOSS development of IT systems using course projects and have not felt discouraged at all by its many challenges. On the contrary!

3.4 Lessons Learned

The learning curve for adopting FOSS practices in computing courses is steep. Working collaboratively and openly with team members, peers, instructor, and other possible contributors requires a significant mindset shift. Being "productively lost" [10] in the classroom, whether one teaches or learns, bears different stakes and risks than those faced when one volunteers to contribute to FOSS and figures out ways to become a productive member.

It has helped me that my first FOSS development teaching experiment was scaled down to less than half of the semester and counted towards less than 25% of the final grade. At the same time, the expectations were set too high. The problems I observed with student learning and development practices were:

- Inconsistent and sporadic use of version control
- Low impact of tutorial materials
- Prevalence of traditional ways of collaboration and little progress with using FOSS collaboration tools effectively.

It became apparent that more emphasis should be given to (1) describing, discussing, and demonstrating what the FOSS development principles and practices are and (2) exposing the consequences of deviating from what is expected of every single member of the FOSS community. Whether a student posts a meeting agenda, commits a SQL script revision, or updates a class diagram in a project wiki, FOSS development activities should be tied to assessing student learning. The old saying "if it's not graded, it doesn't count" bears an undeniable truth. The quality control cycle of FOSS development artifacts should be mapped

directly to how student learning is assessed. To achieve this mapping, the following evaluation cycle should be applied:

1. Team members' roles and responsibilities are examined weekly and openly by students themselves, their peers, and instructors.
2. Responsibilities correlate directly with the artifacts that members are expected to produce. These correlations are also shared openly.
3. Student learning is measured by self-evaluations, peer reviews, and instructor grading of these artifacts.

4. NEXT ITERATION

4.1 Teaching FOSS Development

In Spring 2011 I extended FOSS development in two courses:

- CIS 520 Database Design and Development, the first database course in the CIS program required of all majors and which enrolls primarily sophomores and transfer students at the junior level, and
- CIS 605 Web Application Development, an upper-level elective whose prerequisite is CIS 505, the course I taught in Fall 2010.

A total of 30 students took these courses (18 in CIS 520 and 12 in CIS 605). Student demographics were comparable to those in the CIS 505 course. There were three females (10%), two minority students (6.7%), six veterans (20%), and four students with non-CIS majors (13%). The breakdown by year of study had six sophomores, 14 juniors, and ten seniors.

In the CIS 520 course students were expected to learn how to model, design, and normalize databases, implement their designs in SQL, develop reporting and data entry features for a database application, and design and prototype a business application using a database management system. The course textbooks [11, 12] were available through the library's Safari Books Online Service. Assessment of student learning measured student work in eight weekly homework assignments that used pair reviews and programming (24%), two examinations, using both closed and open texts and notes formats (52%), and a team project with three iterations (24%) that was scheduled during the last six weeks of the semester. The course project component introduced students to FOSS development infrastructure and demanded application of FOSS development practices.

The CIS 605 course was entirely project-oriented. Students were expected to apply advanced web programming concepts and techniques; work with a shared code base among developers with various roles; integrate web applications with database and web server applications; implement, test, document, and deploy a web system; use FOSS practices, development, and collaboration tools to carry out a team project; and communicate timely and work in teams effectively with peer developers and users. I used Tucker, Morelli, and de Silva textbook [13], which is very deliberate about integrating a FOSS process with agile techniques, modern collaboration tools, community involvement, and team work.

To scaffold student abilities and create opportunities for feedback and reviews, weekly homework assignments were used as building blocks towards each of the three project iterations (each of which extended over 3 to 4 weeks, with the first project iteration starting in the fourth week). There were eleven homework assignments (44%), three project iterations (16% each), and a project presentation (including presentation abstract and poster) at the UNH Undergraduate Research Conference (8%).

4.2 FOSS Development Student Competencies

From my experience with these two courses it became apparent that students participate productively and successfully in FOSS development of IT systems if they learn to be:

- Skilled communicators with peer developers, end-users, and clients.
- Adept at working with a shared code base and using version control functions to manage complex code development.
- Active, inquisitive, and prompt participants in discussion threads that frame on-going team conversations hosted by a project mailing list or forum.
- "Compulsive" documenters and skilled technical writers, who know very well the merit of documenting system development and project management activities.
- Experienced with development tools (coding, debugging, unit testing, and automatic generation of code documentation) and familiar with at least one programming language.
- Experienced with building a prototype or release installation and setting the infrastructure where the system's features can be demonstrated and tested.
- Engaged with the IT system's end-users and clients in all phases of the development process, and responsive to their needs as reflected in models, documents, and manuals that "speak" the user's language.
- Familiar with the range of roles developers assume when building and maintaining an IT system, and some exposure to the responsibilities of each those roles, whether analyst, designer, coder, tester, technical writer, IT support engineer, team leader, architect, or reviewer.

5. CONCLUSION

The experience with an open source approach to IT system development in three CIS courses inspired other students and faculty to improve team work activities by using wikis to feature course projects and version control to manage the source code. These practices can be easily adopted by courses in which projects weigh at least 25% of the final grade. The projects can become a learning resource that provides case studies and examples after which other projects can be modeled. They can be also revised and developed further.

Upper-level courses that are project-oriented and the capstone project course are ideal candidates for building a FOSS student community. This community will promote the FOSS culture values of sharing knowledge and skills, mentoring peers, and contributing fixes, features, documentation, manuals, and other artifacts that can improve IT systems at different stages of development and use.

6. REFERENCES

[1] "Growing the Humanitarian FOSS Community". 2010. Second Annual HFOSS Education Symposium, jointly held with SIGCSE 2010 (Milwaukee, OH, Mach 2010).

[2] *HFOSS 2010 Faculty Workshop*. Computer Science Department, Trinity College. (Hartford, CT, May 2010). Retrieved June 3, 2011 from

http://teaching.hfoss.org/index.php/HFOSS_2010_Faculty_Workshop.

[3] *Professors' Open Source Summer Experience (POSSE) Workshop*. 2010. Worcester State University (Worcester, MA, June 2010). Retrieved June 3, 2011 from http://teachingopensource.org/index.php/POSSE_Worcester_State.

[4] Nixon, Robin. 2009. *Learning PHP, MySQL, and JavaScript*. O'Reilly Media, Inc.

[5] Robson, Elisabeth and Eric T. Freeman. 2005. *Head First HTML with CSS & XHTML.* O'Reilly Media, Inc.

[6] Fogel, Karl. 2010. *Producing open source software: How to run a successful free software project*. Creative Commons Attribution Share Alike 3.0. Retrieved June 3, 2011 from http://producingoss.com/.

[7] RMH Homebase 1.5 on SourceForge.net. 2010. Retrieved June 3, 2011 from http://sourceforge.net/projects/rmhhomebase/.

[8] "Women and Information Technology by the Numbers." 2009. National Center for Women and Information Technology. Retrieved June 3, 2011 from http://www.ncwit.org/pdf/BytheNumbers09.pdf.

[9] Johnson, Kenneth M. and Robert Macieski. "Demographic Trends in the Manchester-Nashua Metropolitan Area." 2009. UNH Carsey Institute New England Issue Brief, no. 16, 2009.

[10] DeKoenigsberg, G. et al. 2010. *Practical open source software exploration: How to be productively lost, the open source way.* TeachingOpenSourceway.org. Retrieved June 3, 2011 from http://teachingopensource.org/index.php/Textbook_Release_0.8.

[11] Oppel, A. 2010. *Data Modeling: A Beginner's Guide*. McGraw Hill.

[12] Beaulieu, A. *Learning SQL: Master SQL Fundamentals*. 2009. Second edition. O'Reilly Media, Inc.

[13] Tucker, Allen, Ralph Morelli, and Chamindra de Silva. 2010. *Software Development: An Open Source Approach.* CRC Press.