THE USE OF FREE AND OPEN SOURCE SOFTWARE IN REAL-WORLD CAPSTONE PROJECTS*

Kwok-Bun Yue, Zahabia Damania, Raunaq Nilekani, Krishani Abeysekera
Department of Computer Science
University of Houston-Clear Lake
2700 Bay Area Boulevard
Houston, TX 77058
yue@uhcl.edu, zahabia.damania@gmail.com, access_raunaq@hotmail.com, Abeysekera@uhcl.edu

ABSTRACT

This paper analyzes the use of Free and Open Source Software (FOSS) in 22 real-world graduate computer science projects mentored by industrial partners in the past 2.5 years. It serves as a case study on how FOSS is used in an academic setting as well as by the mentoring companies. A diverse list of FOSS was used, suggesting its ubiquity. The general recognized advantages of FOSS drove their adoption: no licensing cost, flexibility of modification, niche functionality, and perceived growing popularity. Our experience agrees with the perceived advantages of FOSS in engaging students with interesting technologies. It also agrees that the potential to communicate with software professionals was beneficial and sometime inspiring, provided that the students took the initiatives to do so. To provide a complete perspective, the paper also includes a section written from the students' point of view on their experience with FOSS in one project.

1. INTRODUCTION

As Free and Open Source Software (FOSS) becomes increasingly ubiquitous, its role in computer science education swells accordingly. There exists a large collection of papers discussing the experience on using FOSS in individual computer science courses such as database, programming, software design, capstone projects and software engineering [1,2,5,8,10]. There are courses devoted entirely to FOSS such as the course

---

Adaptive and Open Software Systems described in [9], and projects aiming at transforming CS undergraduate curriculum with FOSS culture and methodologies in software development, quality control and project management [3]. There are even works on using FOSS to engage CS students and draw other students to computing majors [4,6].

Some perceived advantages of using FOSS in academic settings include:

- No licensing cost and low overall cost [2,7]
- Flexibility in modification to satisfy needed requirements [7,10]
- The possibility of providing the scale and complexity of well developed and documented industrial strength software [4,8]
- The potential of engaging students with real-world projects with real impact [4,8,9].
- The possibility of interacting with software professionals [4,10]

There are also challenges mentioned in using FOSS that include:

- The need of providing adequate support to students [1].
- Possible lack of documentation or developer's support [10].
- Lack of faculty training in FOSS [4]
- FOSS development not matching the learning outcome goals of a given course [4]

Despite the challenges, there is a general understanding that FOSS provides great opportunities for CS education in higher education and that its rapid adoption will continue [2,4,7,8,10]. The continual publications of results and experience on course designs, best practices and pedagogical considerations of FOSS adoption in CS courses will help facilitating the process. The goal of this paper is to elaborate the experience of the use of FOSS in graduate real-world CS capstone projects. It analyzes the use of FOSS in 22 real-world projects mentored by industrial partners in the past 2.5 years. The paper draws lessons learnt from this analysis to provide a case study on the adoption of FOSS by both the capstone project courses and the mentoring companies. Furthermore, a section written by the student authors of the paper provides insight of using FOSS from the students' points of view.

The paper is organized in the following manner. Section 2 provides a brief background of the capstone project course. Section 3 discusses the results of analyzing the use of FOSS in 22 real-world projects. Section 4 elaborates how FOSS was used in one selected project and what lessons were learnt by the students. Section 5 contains our future direction and conclusions.

2. THE UHCL GRADUATE CAPSTONE PROJECT COURSE

Graduate students of CS and Computer Information Systems (CIS) majors at the University of Houston-Clear Lake (UHCL) must take a capstone project course during their last twelve hours of study. They work as a team of usually four members on real-world projects. There were a few internal projects, but in recent years, nearly all of them were real-world projects mentored by surrounding companies. Students need to satisfy two sets of requirements. The course requirements are specified and assessed by the instructor and include such tasks as project's website maintenance, technical presentations, a technical report, minutes and agendas of meetings, and course-related documentation. The project requirements are specified by the mentors to satisfy the

requirements of the companies. Typical projects are either extension of existing software products or development of prototypes for feasibility study and demonstrations.

Before the beginning of each semester, the industrial mentors develop initial sets of requirement specifications for projects. They are then posted and presented for teams to choose. Projects may mandate the use of specific software. They may also only suggest certain software and allow students to select their own choices. Sometimes, the research on identifying suitable software can be an important component of the project specifications. Thus, teams have varying degree of freedom in selecting software, just like in the real world. Mentors may also require specific software processes and development methodologies that the teams must observe.

We have taught the course in the current format for more than 10 years and accumulated the experience of about 100 real-world projects with industrial mentorship. Based on this experience, the next section discusses an analysis of 22 recent projects as a case study of how FOSS was used in the course as well as by the mentoring companies.

3. FOSS USE IN THE CAPSTONE PROJECT COURSE

A total of 22 real-world projects in five regular semesters from Fall 2008 to Fall 2010 was analyzed. Fourteen industrial partners from seven companies mentored these projects. The backgrounds of three mentors are not in computer science. One of them is the CEO of a small business company and the other two are managers. The remaining eleven mentors are software professionals, including 3 CTOs and 1 CEO. The projects covered a wide range of applications including software process management, workflow engine, bio-medical imaging, enterprise communications, solar power plant, check processing, chemical plant control prototyping, disaster recovery, mobile applications, etc.

In our analysis, we did not count ubiquitous general purpose FOSS such as Linux, Apache, sendmail, etc. We also did not count general productivity software such as Firefox, Open Office, open source text editors, and other small utilities. Furthermore, students might not fully report how they used FOSS to support minor tasks. Thus, the results somewhat undercounted the use of FOSS in these projects.

Out of these 22 projects, 15 (68.2%) of them use FOSS significantly. Four of these projects (18.2%) used FOSS as their centerpiece technologies: Drupal, Android, JForum, eHour and ANTLR (see Table 1 for brief explanation of their purposes.) Three of them (Drupal, Android and eHour) were mandated by the mentors. The other two were selected after team research. The seven capstone projects that did not use FOSS significantly were based on five industrial projects (one project continued for three semesters):

1. A standalone application that used only Microsoft's .NET technology
2. An iPhone's OpenGL application
3. A Graphical Processing Unit application with nVidia's CUDA architecture and C programming
4. A Java-based Web application driven by Microsoft's SQL Server
5. A more research-oriented project on a Web-based no-wait job scheduling problem

Table 1 lists the FOSS used and their frequencies of use in the 15 capstone projects, with brief description of the purposes of the FOSS.

Table 1 Use of FOSS in Capstone Projects

FOSS	# of teams used	Description
MySQL	9	Database Management System
JBoss/Tomcat	7	J2EE Application Server
BlazeDS	5	Java remoting and web messaging technology for Flex
Googlewave	5	Real-time collaboration and communication tool
Eclipse	4	Java IDE
Hibernate	3	Object-relational mapping for Java
Cocoa AMF	2	Objective-C's implementation of the Adobe Flash's remoting format (AMF0/AMF3) for servers and clients
Google Map API	2	API for manipulating Google Map
Liferay	2	Java based open source Web portal platform
VisualText	2	Integrated IDE for text analysis; free for non-commercial use
Android	1	Mobile application OS and development platform
ANTLR	1	ANother Tool for Language Recognition: for constructing compilers and translators
AntTweakBar	1	C/C++ library for light and intuitive GUI for OpenGL and DirectX
Apache XMLRPC	1	Java's XML RPC Server API
Drupal	1	PHP based Content Management System
eHour	1	Web-based timesheet management tool
JForum	1	Java based discussion board
NUnit Testing	1	A JUnit type testing tool for .NET languages such as C#

PHP	1	General purpose scripting language especially for Web applications
VTK	1	Multiple language system for 3D computer graphics, image processing and visualization
ZXing	1	Zebra Crossing: Java's Library for parsing 1-D and 2-D barcodes

Among them, VisualText is free only for non-commercial use and is not open source. Google Map API is free but not open source. The remainders are all open source with varying types of licenses. A total of 21 different FOSS with diverse purposes was used. There were projects where FOSS was keys, but FOSS could also turn up in unexpected ways. For example, an iPhone application used four FOSS in significant ways, and in a pure Microsoft's .NET based project, the team selected NUnit as the testing tool. In projects where technology requirement was not mandatory, FOSS may just be selected as the best option after team research. However, there were also projects where commercial products were eventually selected instead of FOSS alternatives. Overall, the use of FOSS in the capstone projects seems to pick up recently. For example, 9 out of 10 projects in 2010 used an average of 3.5 FOSS significantly. This is quite higher than earlier projects in the analysis.

As expected, workhorses like MySQL, JBoss/Tomcat and Eclipse were frequently used. Five teams used Googlewave in three semesters. Its adoption was helped by the instructor's encouragement as a communications tool in one semester. Student feedback on Googlewave was very positive. Unfortunately, Google announced the discontinuation of supporting the service. The popular use of BlazeDS, a Java remoting and web messaging technology for the Adobe's Flex, is an indicator of mentoring company's interests in Rich Internet Applications (RIA) using Flex/Flash.

The mentoring companies indicated various reasons for them to use FOSS in their mentored projects. Chief among them are the generally agreed advantages of FOSS in low cost and ease of modifications. For example, ZXing was used as an alternative of an expensive commercial product for 2-D barcode parsing in a prototype/feasibility study project even though its performance was inferior in the required barcode format. JForum was used in another Web portal project as the team could modify its code to satisfy specific needs. FOSS may also be used since they were the best available solutions in the niche area of applications. BlazeDS, ANTLR and ZXing are examples of this kind of niche functionality for some projects. Finally, the mentoring companies may also adopt FOSS as they perceive a coming trend of customer demands. The projects that used Drupal and Android fall into this category.

Our experience agrees with the perceived advantages of FOSS in academic settings listed in Section 1. However, only two teams selected to correspond with the development communities and both reported very positive experience in communications with software professionals, a perceived FOSS advantage. They found the FOSS communities to be very responsive in general. Thus, CS educators may consider encouraging students to actively participate in the FOSS community early and often.

For the perceived challenges of FOSS, we did not detect consistent difference between the learning curves of mastering FOSS and commercial technologies. Documentation and support were not an issue for mature FOSS with large communities. However, they can be issues for smaller and less active FOSS projects and they can delay the progress of the capstone projects. Instructors may advise students to take this into their consideration in adopting technologies and to communicate their questions and needs to the FOSS communities as soon as possible.

4. STUDENT EXPERIENCE ON FOSS USAGE IN A CAPSTONE PROJECT

This section is based on the experience of the two student authors of the paper to provide a different lens for understanding FOSS in capstone project. The remaining of this section is written from their first person's viewpoints.

Our team of four students worked on a project jointly mentored by a technology company and a small business. The small business had developed a check, driver license and identity card processing software suite. The software used a commercial product to parse a 1-D or 2-D barcode image into a textual string of stored information. In particular, the 2-D barcode images use a format known as PDF-417. The software used a commercial closed source product to parse PDF-417. However, its license cost is high and as a closed source API, customization is not flexible. The mentors wanted to study and build a prototype based on lower cost alternatives. Thus, a major task of the project was to research and build a prototype as a feasibility study. There were also other important tasks, such as decoding the parsed information into useful details using standards by AAMVA (American Association of Motor Vehicle Administrator), implementing a Rich Internet Application (RIA) Web interface using Adobe Flex/Flash, and the generation of PDF-417 2-D barcodes through a Web application.

In the first half of the semester, with the input from the mentors, our team developed a metric for selecting the PDF-417 parsing products. We then identified three commercial products and a FOSS: Zebra Crossing (ZXing). Our teams conducted testing on the four products and concluded that the three commercial products were significantly better than ZXing in parsing PDF-417 barcodes. However, their license cost structures are comparable to the current product. As closed source API themselves, they do not represent any substantial improvement. ZXing became the only option to the mentors and the metric was abandoned.

While working on other required tasks of the project concurrently, our team then spent much effort in improving the parsing success rate of ZXing using various methods with limited successes,. We started to contact the ZXing development community at the two third point of the semester. The responses were very quick and professional. They were aware of the problems in parsing PDF-417, which was only in alpha version. ZXing supported many decoding standards and fixing the quality of PDF-417 was not among the highest priority within ZXing's active community at the moment. A fix from the development community was not expected to be available soon enough. As time was running out, our team could only implement some peripheral techniques in improving parsing accuracy such as cleaning the input images. The accuracy was less than desirable but the mentors understood the obstacles.

Overall, our experience with FOSS was very positive. The selection process helped us to understand how FOSS may be advantageous but also may be restrictive. In our case, this restriction included accuracy of the parsing result as well as that the language is Java, whereas the existing software used .NET technology and our prototype used Flex/Flash. Additional effort was spent to bridge this incompatibility.

Communications with the ZXing's development team was a great learning opportunity and allowed us to appreciate FOSS culture and software developer's professionalism. In hindsight, we wish that we had communicated with the community much earlier. That would not only have helped us in focusing our remedial actions on the accuracy problem. With only 47 total Java classes and 7 classes on PDF-417 parsing, ZXing is of the right size and complexity for students to make contributions and the community was inviting. Future teams working on the continuation project should consider this option of contributing to the ZXing project.

Our team did participate in FOSS in another way. To generate PDF-417 barcodes, a much easier task than parsing them, we identified an open source Java's implementation and ported it to Adobe's Flash using ActionScript so it can be used in our prototype. In accordance with open source culture, we are in the process of hosting it in the popular open source site Sourceforge.net.

5. FUTURE DIRECTION AND CONCLUSIONS

We presented a case study on the use of FOSS in industrially mentored real-world capstone projects. The rapid adoption of FOSS is expected to continue in both academic and commercial worlds. FOSS provides an opportunity for CS educators to better prepare their students for their future careers. Its culture matches the academic setting very well. We will continue to experiment with FOSS actively and refine best practices in adopting FOSS. Our team is also working on a meta-analysis in providing a holistic view for the many literatures that are now available in the use of FOSS in computer science and information systems.

REFERENCES

[1] Carrington, D. & Kim, S., Teaching software design with open source software. *33rd ASEE/IEEE Annual Conference on Frontiers in Education*, 5-8, November 2003.

[2] Corbesero, G., Rapid and inexpensive lab deployment using open source software. *Journal of Computing Sciences in Colleges*. 22, (2), 228-234, December 2006.

[3] Dionisio J., Dickson, C., August, S., Dorin, P. & Toal, R., An open source software culture in the undergraduate computer science curriculum. *SIGCSE Bulletin*, 39, (2), 70-74, June 2007.

[4] Hislop, G., Ellis, H., Tucker, A. & Dexter, S., Using open source software to engage students in computer science education. *Proceedings of the 40th ACM technical symposium on Computer science education* (SIGCSE '09), 134-135.

[5] Liu. P., Using open-source robocode as a Java programming assignment. *SIGCSE Bulletin*, 40, (4), 63-67, November 2008.

[6] Morelli, R., Tucker, A., Danner, N., De Lanerolle, T., Ellis, H., Izmirli, O., Krizanc, O. & Parker, G., Revitalizing computing education through free and open source software for humanity. *Communications of the ACM*, 52, (8), 67-75, August 2009.

[7] O'Hara, K. & Kay, J. 2003. Open source software and computer science education. *Journal of Computing Sciences in Colleges*, 18, (3), 1-7, February 2003.

[8] Raj, R. & Kazemian, F., Using Open Source Software in Computer Science Courses, *36th ASEE/IEEE Annual Conference on Frontiers in Education Conference*, 21-26, October 2006.

[9] Seiter, L., Computer Science and service learning: Empowering nonprofit organizations through open source content management systems, *Humanitarian FOSS Project Symposium*, 2009.

[10] Yue, K., De Silva, D., Kim, D., Aktepe, M., Nagle, S., Boerger, C., Jain, A. & Verma, S., Building Real World Domain-Specific Social Network Websites as Capstone Projects, *Journal of Information Systems Education*, 20, (1), 67-76, Spring 2009.