Considerations for Adapting Real-World Open Source Software Projects within the Classroom

Hyunju Kim
Department of Mathematics and Computer Science
Wheaton College
Wheaton, IL, USA
e-mail: hyunju.kim@wheaton.edu

***Abstract*—As Open Source Software (OSS) has become one of the main approaches for developing new software products, efforts to incorporate real-world OSS projects into the computer science classroom have increased. This paper reviews such efforts and discusses the benefits and challenges of adapting OSS projects in software development or engineering courses. It also presents considerations for selecting and using OSS projects for in-classroom software development.**

Keywords-open source software; OSS; software engineering education.

I. Introduction

Open Source Software (OSS) has been widely used in many areas and has become one of the main approaches for developing new software products. As a result, efforts to adapt OSS and its community structures in computer science education have also increased. Findings from such efforts show that using OSS to teach various aspects of software engineering benefits students by providing opportunities for real-world software development, software reengineering, and team activities, such as project management and group communications. These opportunities are rarely available within the traditional classroom environment; thus, adapting OSS in software engineering education can be an effective supplemental teaching approach to prepare students for their future careers.

Section II of this paper will review previous efforts to use OSS projects in the computer science classroom, along with benefits of using OSS components in software engineering education. Section III will present two different approaches to utilizing real-world OSS projects for software development education, and the corresponding challenges and considerations based on our classroom experience. Section IV concludes our experience and discusses future work that will be necessary for better utilizing real-world OSS projects in the classroom setting.

II. Previous Studies

Several studies have already adapted real-world OSS projects to the computer science classroom environment. A study by Hislop et al. [3] identified the effects of adapting Humanitarian Free and OSS projects in undergraduate software engineering or OSS development courses at diverse educational institutions. Findings from the study indicate that students were motivated by participating in such OSS projects and learned various aspects of collaborative software development. Similarly, a study by Stroulia et al. [8] reported on the Undergraduate Capstone Open-Source Project, which offered a distributed software engineering course to students from multiple universities. The course asked students to work on existing, active OSS projects so that they could learn and participate in real-world team activities for developing software. This distributed environment was helpful for students in learning communication skills, as well as in learning from others and through examples. Another study by Krogstie [4] reported the roles and benefits of a broker between a student development team and an OSS community in a senior project course. While working as the gatekeeper, the broker strengthened the programmer’s authority within the team and increased the communication credibility of both parties. As a result, the broker’s role became significant for the student team in acquiring the necessary development knowledge from the OSS community.

While most of the previous efforts have introduced OSS projects to the senior level of computer science studies, a couple of them have incorporated OSS projects into second or third year studies [5][6]. Students were asked to contribute to active OSS projects, so they might learn software evolution processes, such as reverse engineering and software maintenance. Students’ feedback from those courses was positive and reflected their having learned values of documentation, software development tools, and communication with real-world developers.

It seems obvious that incorporating real-world OSS projects into the classroom provides valuable opportunities for students to learn technical and social aspects of software development, such as:

- Communication skills
- Project management activities
- Distributed software development tools
- Problem analysis and solution development according to given constraints
- Learning from others and by example

However, despite these benefits, one of the challenges is to identify OSS projects that are appropriate for student development. A study by Smith et al. [7] initially considered programming language, code size, team development, and buildability as the criteria when choosing appropriate OSS projects to teach software engineering. Later, the study also considered additional criteria, such as the project's application domain, modular design, recent activity, and documentation quality. Results from this study showed that it was not easy to find appropriate projects of proper size. In addition, buildability problems existed among small, single developer projects; levels of documentation varied; and a project's code organization could mislead its code modularity.

This paper reports findings from adapting two different types of OSS projects into software development courses. We will also discuss the pros and cons of the two adaptation forms so that they can serve as criteria for selecting future OSS student development projects.

III. Utilizing OSS Projects within the Classroom

According to Black Duck's 2015 Future of Open Source Survey [1], over 65% of the companies surveyed took an OSS-first approach in developing or using software for daily business. The 2016 survey revealed that about 65% of the companies contributed to OSS projects to influence OSS markets, and 59% of respondents participated in OSS projects to gain competency [2]. Computer Science education must not only respond to these trends and needs, but also integrate learning opportunities that will prepare students for the contemporary corporate environment. OSS projects provide such opportunities in the following areas:

- Community activities: OSS projects require participants to take on diverse roles, such as end-users, project managers, programmers, and testers. Thus, all levels of computer science students can participate in the community activities, which provide a learning area for students according to their interests and knowledge levels. As members of the community, they can learn from others, including real-world professionals. Students will also learn how to effectively work and communicate with others and how to follow community rules and standards.
- Project management: OSS projects are continuously evolving. Thus, students can witness and participate in ongoing activities for forking and merging projects, release planning, code repository management, risk management, and quality management.
- Software reengineering or reverse engineering: Traditional software engineering courses usually focus more on forward engineering. On the other hand, OSS projects require students to understand existing code, including algorithms, software architectures, data structures, and documentation. The code may include good and bad coding practices, and thus students can learn through examples. This will also be a practical teaching resource for software reengineering.
- Communication and development tools: Although traditional computer science labs provide hands-on exercises with tools, they are usually limited in terms of type and scope. Real-world OSS projects can expose students to diverse and cutting-edge documentation, builds, version controls, and testing tools.

A. *Adapting Two Types of OSS Projects*

Wheaton College is a liberal arts college, and its undergraduate enrollment is about 2,400. In 2017, the computer science program at Wheaton offered a software development course to sophomores and juniors and an OSS development course to juniors and seniors. Both courses were offered to computer science majors and minors. The software development course was required for computer science majors, and the OSS development course was offered as an elective, project course. The sizes of the classes were twelve and three respectively.

Students in each course worked on OSS projects of their choice. Those in the software development course worked on projects that were forked from an existing OSS project, and each of the three students in the OSS development course participated in a different, active OSS project. This paper presents preliminary findings based on the student course evaluations and the instructor's observations. The course evaluations included survey questions about their experiences on OSS projects. Because of the small class sizes, the instructor was also able to closely observe students' project activities and their interactions with the existing projects.

As mentioned, there were two different types of student OSS projects. Students either joined active OSS projects or initiated their own OSS projects by forking existing ones. The former type provides the various learning opportunities outlined above, while benefits of the latter type were identified as follows:

- Students have full control over the projects. They can execute the projects according to their own pace and set their own rules and standards for project activities.
- Students can become involved in various management activities. First-time OSS participants, especially student participants, can hardly contribute to management tasks in a large, active OSS project due to their lack of reputation, knowledge, and experience. However, this type of OSS project allows students to practice the full set of management tasks.
- Students can better exercise reengineering activities. This type of OSS project is relatively small, and thus students can understand the existing code better and more quickly. Consequently, the quality of the outputs from refactoring and documenting the code can be improved.

Despite these benefits, initiating a new OSS project may not provide the full benefits of joining preexisting OSS projects because, in a new project, students' interactions are limited to themselves. Thus, it is necessary to consider the pros and cons of both project adaptation types according to students' needs and constraints.

B. *Considerations for Using OSS Projects*

As previously discussed, one of the challenges in utilizing OSS projects is identifying appropriate OSS projects for students. Feedback from our two courses shows that the following criteria should be considered when selecting OSS projects:

- Personal interest: Students indicated that a significant factor motivating them the most was their personal interest. Students were interested in particular applications, technologies, systems, subjects, or programming languages. They preferred choosing OSS projects freely rather than choosing ones from a list of pre-selected OSS projects. Maintaining their interest while working on the projects was considered one of the keys to having successful projects.
- Community-related aspects: Project popularity and status (i.e., active or inactive) should be considered. These aspects can be measured by considering the number of developers, weekly downloads, and the organization of project websites.
- Software product-related aspects: Depending on course requirements and students' interests, programming language, lines of code, platform or system, development tools, code modularity, and documentation can be used to screen projects for students.

We found that prompt and helpful responses from the OSS project's existing developers highly encouraged students to become deeply involved in the project's activities. Those students in the OSS development course started to collaborate with the real-world developers, and their contributions to the projects were reviewed by the developers and adapted by the projects. Such collaborative activities encouraged the students to keep working on the projects even after the semester ended. Thus, the criteria to measure community activities should be thoroughly considered. Among the product-related aspects, code modularity and documentation are essential factors for estimating software quality. Yet, it is difficult to automatically measure modularity and documentation levels. Students should be guided to carefully examine these two aspects while investigating potential OSS projects.

Another challenge is the steep learning curve during the early phase of an OSS project. Students need to handle development and/or build tools and set up the specific environment that the development requires. Those tools may be completely new to students, and the project may not provide full instructions for building and configuring the system. Software tools that are frequently involved in the early phase include IDE, version control systems, build tools, automatic testing tools, and bug/issue trackers.

For instructors, assessing students' contributions to the OSS project is a challenging task as well. When real-world projects are brought to the classroom, the traditional methods for evaluating student performance may not work properly. Instructors then need to utilize information and data from the project's version control system and communication tools to evaluate the students' performances. At the same time, the types, scopes, and difficulties of tasks should be considered. Therefore, developing and presenting a rubric for code commits, documentation, and communication activities will be helpful in establishing course expectations.

IV. Conclusion

This paper presents the considerations, benefits, and challenges of adapting real-world OSS projects to software development courses. The use of OSS projects within the classroom can be a good supplement to traditional approaches for teaching software development. Despite the aforementioned challenges, OSS projects provide various opportunities for computer science students to explore and learn new technologies according to their own interests. OSS projects also allow students to take their knowledge from the classroom and apply it to real-world experiences.

Student feedback from our courses shows that participants gained a significant amount of knowledge from different projects and people. Working on an OSS project helped them build a comprehensive understanding of software development, and contributing to real-world projects was highly rewarding for them.

However, to better utilize real-world OSS projects in the classroom, the following future work should be done:

- Criteria must be carefully considered to select appropriate OSS projects according to course requirements and student interests.
- Instructors should formulate guidelines to help students cope with technical difficulties involved in real-world development activities.
- Student performance evaluation criteria and procedures must be specifically developed for non-traditional, real-world, interactive activities.

We will keep utilizing OSS projects as supplementary teaching tools for the software development course as well as other related courses. Student feedback and data form the courses will be used for the future work.

As another approach to better exploit OSS projects within computer science courses, OSS components can be introduced during the early stages of computer science studies. This will encourage students to continuously work on OSS projects according to their interests and knowledge levels, and contributions to the projects will become a great portfolio of their software development activities.

References

[1] Black Duck, *2015 Future of Open Source Survey Results* [Online]. Availave from https://info.blackducksoftware.com/web-future-of-open-source-LP.html, 2018.08.07

[2] Black Duck, *Future of Open Source Survey 2016 Results* [Online]. Available from https://www.brighttalk.com/webcast/13983/199027, 2018.08.07

[3] G. Hislop, et al., "A Multi-institutional Study of Learning via Student Involvement in Humanitarian Free and Open Source Software Projects", Proc. ICER 2015, 2015, pp. 199-206.

[4] B. R. Krogstie, “Power through Brokering: Open Source Community Participation in Software Engineering Student Projects”, Proc. ICSE 2008, 2008, pp. 791-800.

[5] R. Marmorstein, “Open Source Contribution as an Effective Software Engineering Class Project”, Proc. ITICSE 2011, 2011, pp. 268-272.

[6] R. McCartney, S. Gokhale, and T. Smith, “Evaluating an Early Software Engineering Course with Projects and Tools from Open Source Software”, Proc. ICER 2012, 2012, pp. 5-10.

[7] T. Smith, R. McCartney, S. Gokhale, and L. Kaczmarczyk, “Selecting Open Source Software Projects to Teach Software Engineering”, Proc. SIGCSE 2014, 2014, pp. 397-402.

[8] E. Stroulia, K. Bauer, M. Craig, K. Reid, and G. Wilson, “Teaching Distributed Software Engineering with UCOSP: The Undergraduate Capstone Open-Source Project”, Proc. CTGDSD 2011, 2011, pp. 20-25.