A Framework for Teaching Software Testing using F/OSS Methodology

Sulayman K Sowe[1], Ioannis Stamelos[1] and Ignatios Deligiannis[2]

[1] Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece. Tel: +30-2310-991927 Fax: +30-2310-998419
sksowe@csd.auth.gr, stamelos@csd.auth.gr
[2] Information Technology Department, Technological Education Institute, 54700 Thessaloniki, Greece. igndel@it.teithe.gr

Abstract. In this paper we discuss a framework for teaching software testing to undergraduate students' volunteers. The framework uses open source software development methodology and was implemented in the "Introduction to Software Engineering" course at the department of Informatics, Aristotle University, Greece. The framework is in three phases, each describing a teaching and learning context in which students get involved in real software projects activities. We report on our teaching experiences, lessons learned and some practical problems we encountered. Results from preliminary evaluation shows that students did well as *bug hunters* in the bazaar and are willing to participate in their projects long after graduation.

1 Introduction

Software engineering (SE) educators are always in search of relevant materials and novel pedagogies that will provide life-long learning experiences and improve the quality of students learning outcomes. However, the teaching and learning situation in SE courses in most universities is acute. Students do not get the chance to participate in long-term projects where they can be exposed to the SE principles and techniques we teach them. In most cases students have to complete their assigned projects in one semester, making it difficult for them to be involved in large and long-term projects. The reality is that SE education does not always expose students to "real-world" projects [3]. Involving students in software projects in local companies is one way of exposing them to real software projects. However, [7] concluded that most companies are not willing to sacrifice their software to students. By utilizing Free and Open Source Software (F/OSS) projects freely available in the Internet, computer science (CS) lecturers may overcome this obstacle. F/OSS projects are '*bazaars of learning*'- they offer a meaningful learning context in which students can be exposed to real-world software development. In this paper we present a framework which provides such a context. The framework was implemented as a pilot program to teach *software testing* in the Introduction to Software Engineering (ISE) course. Fifteen undergraduate students took part in the program. Our evaluation of the

Please use the following format when citing this chapter:

Sowe, S.K., Stamelos, I., and Deligiannis, I., 2006, in IFIP International Federation for Information Processing, Volume 203, Open Source Systems, eds. Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G., (Boston: Springer), pp. 261-266

framework shows that students did well in software testing in F/OSS projects. Our contribution may also strengthen some areas of the IEEE/ACM CS curriculum guidelines [4], which recommends that a CS curriculum should incorporate *Capstone projects.* Like F/OSS projects, capstone projects are managed by the students and solve a problem of the student's choice.

F/OSS in Software Engineering Education: The Bazaar model [5] of developing F/OSS represents a decentralized software development where volunteers develop software online, relying on extensive peer collaboration through the Internet. In F/OSS projects, the developer-user alliance exposes the source code to a large number of testers and ensures rapid evolution of the code. According to Linus Law ("*Given enough eyeballs all bugs are shallow*"), many people (testers, debuggers, co-developers) looking at the source code will ensure that bugs/defects will be found and fixed quickly. Many studies (e.g. [1, 2, 3, 6]) see F/OSS as a pedagogical tool and a viable methodology which gives students practice in dealing with large quantities of code written by other people. Important as these studies are, a framework for teaching SE courses in general and software testing in particular in the informal context of F/OSS is lacking in the literature. The overlook might be that the F/OSS paradigm has some peculiar characteristics which make teaching in this context harder to integrate into the formal SE curricular structure of most universities.

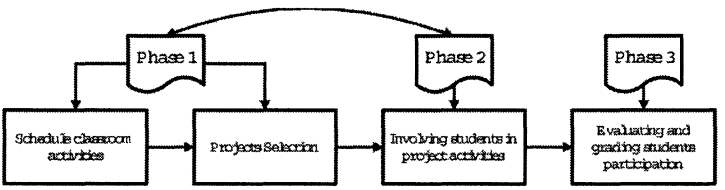

Fig. 1. F/OSS Teaching and Learning Framework.

2 F/OSS Framework for Teaching Software Testing

The Introduction to Software Engineering (ISE) course, in which the framework was implemented, is offered as a 12 weeks course during the 5^{th} semester. The F/OSS framework for teaching software testing is shown in Figure 1.

2.1 *Phase 1*

At the beginning of the semester we discussed with the 150 students enrolled in the course about involving them in software testing in F/OSS projects. Fifteen students volunteered to take part. For the first two weeks the students received 8hrs of lectures on the following topics:

- *F/OSS project* (Activities and Testing in F/OSS).
- *F/OSS communities* (Formation and Roles).
- *Communication* (Etiquettes of forums/mailing lists).
- *Collaborative platforms* (CVS, Bugzilla, Bug Tracking Systems (BTS), etc).

At the end of the session students were guided to browse projects hosted at *sourceforge.net*. In choosing a project, students were asked to pay particular attention to the following *F/OSS projects selection criteria*:

- operating system (Linux, Windows) and programming language used,
- number of developers and how active the forums are,
- development status (Alpha, Beta, Mature).

Having identified their projects, each student was asked to make a class presentation, detailing the history of the project, bug reporting procedures, and testing tools used.

2.2 *Phase 2*

In week 3 students learnt how to register in their projects, use bug tracking systems, and browse and report bugs. They practiced writing fictitious bug reports for their colleagues to criticize. In their projects, students implemented the testing strategy shown in Figure 2.

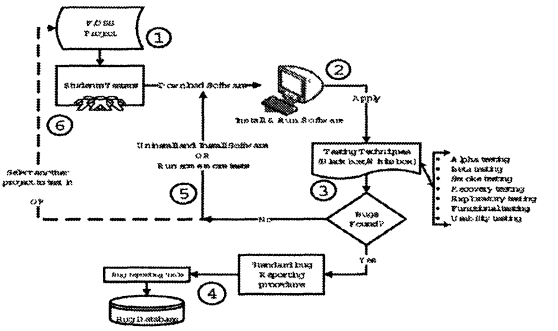

Fig. 2. F/OSS Testing Strategy.

They download and installed the software (**1** - **2**) and applied various software testing techniques (**3**). This may result in the discovery of bugs, which are then logged into the project's bug database using standard bug reporting procedure and tools (**4**). Where a student is not able to find a bug, he/she may run more tests (**5**) or selected another project to continue testing (**6**). Every time a student submitted a bug, he/she notifies the lecturer. Students were asked to continuously login to check the status of their submission and engage community members. During the fifth week students who already made progress in

their projects by finding and submitting 2-3 bugs were asked to make a class presentation to discuss their experiences (e.g. types of bugs found, how they were found, what they think caused the problem, how they reported them, and what responses, if any, were received).

2.3 *Phase 3*

Based on their presentations and testing activities, the students were graded as follows: Class presentation (10%), Project participation (12%), Concise bug reports (13%), and Testing activity (15%).

3 Results and Discussions

In validating the framework we discuss students' participation in their respective projects and the results of a survey we conducted.

3.1 Students Testing Activates

At the end of *Phase 2*, two students withdrew from the program. The remaining 13 tested in 16 projects[1], found 72 bugs, reported 68, fixed 15, and received 43 replies from the F/OSS community. The mean numbers of bugs found and reported per student were 5.54 and 5.23, respectively. This means that students reported slightly less bugs than they found, because some of the bugs they found were already reported. The mean value of bugs fixed per student was 1.15. Thus, the students performed best in finding and reporting bugs in their projects. They did not do well in fixing bugs. The mean number of responses to a bug report was 3.31. Figure 3 shows how the students fair in each activity.

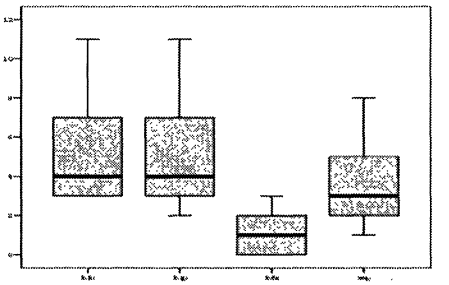

Fig. 3. Distribution of students' testing activities. (***bfn***)=bugs found. (***brp***)=bugs reported. (***bfx***)=bugs fixed. (***rep***)=replies to a submitted bug.

[1] Games (8), Mozilla Suite (4), Multimedia (2), Mobiles and Networks (1), Astronomy (1)

3.2 Survey results

In week 6 the students were invited to complete an online questionnaire containing 21 items. The aim of the survey was to validate the framework from students' point of view. Ten students completed the survey. We group the responses into five categories.

1. Students Motivation. According to the survey students enjoyed software testing in F/OSS projects (100%) and would continue testing in their projects after graduating (90%). Furthermore, most students would prefer to have their other CS courses taught using F/OSS methodology (90%).
2. The Teaching Context. 80% of the students reported getting help from the lecturer when selecting their projects, making it easy for 60% of them to find a project to participate in. While students collaborated and discussed their projects amongst themselves (90%), 80% preferred discussing their projects and bug reports (50%) with the lecturer.
3. Using F/OSS Testing Tools. 80% of students prefer the BTS to report bugs because it is easy to use.
4. Testing Activity. On average, students used the software for at least 1-2 days (50%) before they could find any bugs. Since students found the BTS easy to use, the process of reporting bugs was also easy (90%). While it was easy for most students to describe the bugs they found (70%), 20% found this exercise difficult. When asked if finding bugs in their projects was easy, students responses were evenly split (50% - 50%). Students were able to read and understand bugs others reported (80%), but only a few (30%) are able to fix any bugs reported in their projects. Even a smaller percentage (20%) were able to fix their own bugs. So our students could best be described as *bug hunters* than bug fixers. In this role students are able to contribute to their projects 'eyeballs' just looking for and contributing bugs.
5. F/OSS Community Response. At the beginning many students were hesitant that they were not getting prompt feedback, but 70% of them later reported that their projects' communities are very responsive. 60% reported that their projects (or rather the portals which host the project) provided useful information to help them in their bug reporting activity.

4 Conclusion

In this paper we have proposed a framework for teaching software testing using F/OSS methodology. The implementation of the framework in a formal CS course with a sample of fifteen undergraduate volunteers was discussed. Our experience shows that SE education could benefit from such a teaching and learning approach by exposing students to "real-world" software engineering projects. The projects in which the students tested were very responsive and appreciative. While we have already graded and published the students results, we still continue to get emails from them about responses they received from their

projects. We enthusiastically continue to respond accordingly. Our presentation resolves two key issues about F/OSS in SE education. Firstly, project-based CS courses need not depend on closed-source projects outside the university in order to give students experience in real-world projects. Second, it is possible for CS lecturers to integrate the informal F/OSS teaching and learning context into their formal curricular structure to teach CS courses (e.g. software testing). It was satisfying to note that most students will continue participating in F/OSS projects after the end of our pilot program. However, we were faced with the hard reality that students must complete their testing activity at the end of the semester.

Validity threats and future work: Our data set consists of a small random sample of student volunteers, about 10% of the students in the ISE course. Thus, there is danger in generalizing the results to other CS courses, classes, and possibly to other universities, where sample size, skills, and backgrounds of the students are probably different. However, because there are few published results in this area, we hope that our findings will act as a base for further research in this area. We plan to repeat the program with a larger sample next semester. Furthermore, we are currently conducting two online surveys (post-students survey and staff survey) to help us further validate the framework.

References

1. D. Carrington, and S. Kim, Teaching Software Engineering Design with Open Source Software.*33rd ASEE/IEEE Frontiers in Education Conference*,(May 16, 2005); `http://www.cs.wm.edu/~coppit/csci690-spring2004/papers/1273.pdf`
2. M. D. German, Experience teaching a graduate course in Open Source Software Engineering. In P*roceedings of the first International Conference on Open Source Systems*. Genova, 326-328 (2005)
3. C. Liu, Enriching software engineering courses with service-learning projects and the open-source approach. In *Proceedings of the 27th international Conference on Software Engineering*, ICSE '05. ACM Press, 613-614 (2005).
4. Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, *IEEE/ACM Joint Task Force on Computing Curricula*, (2004), (December 10, 2005); `http://sites.computer.org/ccse/SE2004Volume.pdf`
5. S. E. Raymond, *The Cathedral and the Bazaar*. O'Reilly, Sebastopol,(1999).
6. S. K. Sowe, A. Karoulis, and I. Stamelos, A Constructivist View of Knowledge management in Open Source Virtual Communities. In A. D. Figueiredo, A. P. Afonso (Eds), *Managing Learning in Virtual Settings: The Role of Context*. Idea Group Inc., 285-303 (2005).
7. Z. Alzamil, Towards an effective software engineering course project. In *Proceedings of the 27th international Conference on Software Engineering, ICSE '05. ACM Press*, 631-632 (2005).