# Involving Software Engineering Students in Open Source Software Projects: Experiences from a Pilot Study

**Sulayman K. Sowe**
**Ioannis G. Stamelos**
Department of Informatics
Aristotle University
54124 Thessaloniki, Greece
sksowe@csd.auth.gr , stamelos@csd.auth.gr

**ABSTRACT**

Anecdotal and research evidences show that the Free and Open Source Software (F/OSS) development model has produced a paradigm shift in the way we develop, support, and distribute software. This shift is not only redefining the software industry but also the way we teach and learn in our software engineering (SE) courses. But for many universities F/OSS is seen as an optional low cost technology to support the IT infrastructure and administrational duties. Few see F/OSS as an opportunity for students to learn the SE concepts and skills we teach. Furthermore, it is still an open question as to whether the F/OSS methodology can be effectively used to teach SE courses within the formally structured curriculum in most universities. This paper discusses F/OSS projects as *bazaars of learning* that offer a meaningful learning context. The discussion is centered on a pilot study in which students were involved in software testing in F/OSS projects. We present the teaching and learning framework we used in the pilot study and report on our experiences, lessons learned, and some practical problems we encountered. Our grading and evaluation approach show that the students did relatively well as bug hunters and reporters. Results from two online surveys indicate that students are motivated in participating in software testing in the bazaar, and they are willing to participate in project activities long after their graduation. The study reveals one possible way SE educators can teach and integrate F/OSS into their formal curricular structure.

**Keywords:** Software Engineering Education, Open Source Software Projects, Capstone Projects, Software Testing, Learning Objectives, Teaching and learning.

## 1. INTRODUCTION

Software engineering (SE) educators often emphasize that SE courses should have a significant 'real-world' experience necessary to enable effective learning of software engineering skills and concepts. However, students still graduate from universities without getting the chance to participate in realistic and long-term SE projects (Liu, 2005). One reason for this might be due to the fact that, in SE projects, the real-world involves participants with different skills and experiences and is full of inconsistencies, complex, and changing all the time (Hans, 2005). Thus, getting students involved in such a complex environment while at school is not only challenging for students and instructors, but also difficult to implement in the formal teaching and learning structure of most SE courses.

The joint IEEE/ACM CS curriculum guidelines (IEEE/ACM, SE2004) suggest that CS curricular should:

- have a significant real-world basis necessary to enable effective learning of software engineering skills and concepts,
- incorporate Capstone projects. Students need a significant project, preferably spanning their entire last year, in order to practice the knowledge and skills they have learned.

Many efforts, in terms of teaching and research, have been made with regards to these guidelines. For example, Alzamil (2005) demonstrated that involving students in software projects in local companies is one way of effectively teaching SE courses. But he concluded that most of these companies are not willing to sacrifice their product quality to students.

Free and Open Source Software (F/OSS) development not only exemplifies a viable software development approach, but also a model for the creation of self-learning (Sowe, *et. al.*, 2004) and self-organizing communities. Enable by the Internet, geographically distributed individuals voluntarily contribute to a project by means of the Bazaar model (Raymond, 1999). Extensive peer collaboration allows project participants to write code, debug, test, and integrate software. Communities in various projects provide support services such as suggestions for products features, act as distributing organs, answer queries, and help new members having problems with the software. Research evidences suggest that the communities in various F/OSS

projects provide free help or 'user-to-user assistance' (Lakhani and Hippel, 2003) and problem solving for participants. Such communities are also focus on learning and the sharing of knowledge (Holtgrewe, 2004, Sowe, *et. al.*, 2006c).

In recent times, F/OSS is making inroads not only in business and software industries but in colleges and universities as well. There is increased interest in the F/OSS learning environment (Sowe, *et. al.*, 2004) and in F/OSS projects as *bazaars of learning* (Sowe, *et al.*, 2006a). F/OSS is both an alternative teaching methodology and an educational model (Faber, 2002). Computer science students can be involved in meaningful software development activities and get experience in dealing with realistic software systems with large quantities of code written by other people (Carrington and Kim, 2003). Many universities have also started teaching F/OSS course (German, 2005; Megias, *et al.*, 2005; Ozel, *et. al.*, 2006). Projects (e.g. Edukalibre) have been launched to study the transfer of F/OSS practices to the production of educational resources (Barahona, *et. al.*, 2005). Another European Union funded project, FLOSSCom studies how the principles of F/OSS communities can be used to improve ICT supported formal education. Workshops (e.g. tOSSad) have also started discussing the adoption of F/OSS in education (Ozel, *et. al.*, 2006).

These studies show that, pedagogically, software engineering educators may utilize F/OSS to extend the methodology by which we learn, apply, and teach SE courses. However, the F/OSS projects environment is different from the formal SE teaching and learning context in many colleges and universities. Important as these studies are, they fail to address the *challenges* software engineering educators face. For example, how to teach SE courses using F/OSS methodology in the formally structured SE curriculum.

Furthermore, with Capstone projects, students must be able to meet some learning objectives of a typical SE curriculum:

OBJ1: Show mastery of the software engineering knowledge and skills, and professional issues necessary to begin practice as a software engineer.

OBJ2: Work as an individual and as part of a team to develop and deliver quality software artifacts.

OBJ3: Demonstrate an understanding and appreciation for the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment.

OBJ4: Learn new models, techniques, and technologies as they emerge and appreciate the necessity of such continuing professional development. (IEEE/ACM, SE2004, pp.15-16)

Even with Capstone projects many SE engineering courses still face problems meeting some of these learning objectives. F/OSS projects as bazaars of learning may not be the cure for this ill but may go a long way in meeting some of these objectives.

In this paper we present an F/OSS teaching and learning framework which addresses the challenge SE educators face and how best some of these learning objectives can be met. The framework for teaching SE courses in general and software testing in particular was implemented as a pilot study at the Department of Informatics, Aristotle University. The aim of the study is to:

- provide opportunity for our students to work on what they considered interesting themselves,
- give the students real-world experience in dealing with large software projects.

In the pilot study, students volunteered, just as every F/OSS developer does, and had to choose their projects. We only wanted to provide them with useful guidance and support. Furthermore, we hope that our experiences in this study will lead us to further experimentation with a larger group of students.

The rest of the paper is organized as follows. Section 2 introduces the F/OSS framework and discusses what each phase entails. Section 3 presents the grading and evaluation approach we used. This section uses results of students' participation in their projects and their responses to the two online surveys. Section 4 lists experiences and lessons learned in the implementation of the framework. A discussion on how well the pilot study seemed to meet the learning objectives of Capstone projects and some validity threats to our study are presented in Section 5. Our concluding remarks and future research are presented in Section 6.

## 2. F/OSS FRAMEWORK FOR TEACHING SOFTWARE ENGINEERING COURSES

The piloted F/OSS framework was implemented within the teaching and learning context of the Introduction to Software Engineering course (ISE) and lasted approximately 12.5 weeks. ISE is one of the 72 undergraduate courses offered by the department of informatics. The course is compulsory for computer science majors and is offered as a 12-13 weeks course during the 5th semester. The objectives of the course are twofold; to provide students with a "pragmatic picture of software engineering research and practice" (Pfleeger, 1998), and expose them to the principles software engineering as a laboratory and practical science. In the ISE course students have 2hrs/week lectures and 2hrs/week of laboratory work. As part of their assignments students work in small groups, writing and execute test plans for their group projects. Topics covered in the course range from software development models and process, project planning and management, system design, software maintenance, etc, to testing individual programs and complete systems. The topic of interest to us in this paper is software testing. Some of the courses students would have completed prior to the ISE course are:

- Semester 1: C language (Basic Constructs)
- Semester 2: Advance C language, UNIX
- Semester 3: C++, Logic and Functional Programming
- Semester 4: PROLOG, Compilers

During semesters 1 to 4 students would have acquired certain software development skills (liu, 2005) which may be vital to the software testing aspect of the ISE course and the implementation of our framework. These skills are:

- Writing small programs (usually in C language) as their programming assignments
- Developing software in teams and collaborating in small-scale software projects.

In the ISE course we try to make students understand the difference between testing the small programs they write for themselves in class and as assignments and the testing of large scale software products that they might deal with when they graduate. The teaching and learning context focuses on the identification of software faults and failures, unit and integration testing, function and performance testing, writing and execution of test plans/cases, etc.

Two lecturers were involved in the pilot study. One was responsible for scheduling F/OSS activities, the other acting as an adviser. Students at their previous semesters have already been taught programming, so coding is not a focal point of the ISE course. Instead, focus is placed on other activities such as software testing.

The framework shown in Figure 1 is in three phases. Each phase describes a context in which students get involved in F/OSS projects activities. Their involvement was basic. Students select a project and download and use the software. Any problems they encounter in the use of the software are reported to the project's community for action. Their main tasks were to find and report bugs in their respective projects. These tasks may take the form of functional, usability, or smoke testing. In what follows, we discuss each phase in turn.

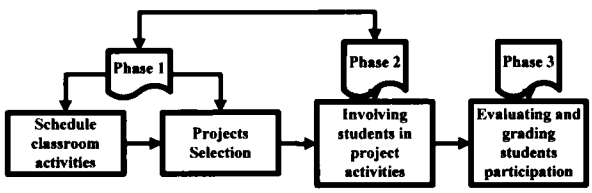


**Figure 1. F/OSS Framework for Teaching Software Engineering Courses, (Sowe *et al.* 2006a; pp.262).**

### 2.1 Phase 1

Phase 1 was a preparatory stage in which the lecturer scheduled classroom activities and guided the students in their project selection. We discussed with the 150 students in the ISE course about involving them in software testing in F/OSS projects. Fifteen students volunteered to take part in the pilot program, but only thirteen students completed the exercise. The F/OSS development process is different from traditional software development that students are taught in their CS courses. Thus, it's vital at this phase that students are introduced to F/OSS. Our introductory lectures were on the following topics:

- **What is F/OSS?** This section covered the F/OSS development process, activities in projects, the rights various licenses (e.g. GNU/GPL) grants the user of F/OSS, etc.
- **F/OSS communities:** Formation, structures and members' roles. We discussed communities in the Linux, Apache, and Debian projects.
- **Communication:** We discussed etiquettes of forums, mailing lists (moderated and un-moderated), and Internet Relay Chats (IRCs).
- **Collaborative platforms:** We introduced the students to CVS, Tinderbox, Bugzilla, bug tracking systems (BTS) and how to browse bug databases.

At the end of the introductory lectures the students were guided to explore sourceforge.net, a repository of F/OSS projects. This session was intended to give the students a feel as to the category of F/OSS projects available on the Internet. At the end of the *exploratory process* the students selected their projects. In choosing a project, the students followed these F/OSS projects *selection criteria.*

- **Operating system/platform** (Linux, Windows, etc). Students may choose projects which run on platform they are most comfortable with.
- **Size of ownership/developers.** According to the Bazaar model (Raymond, 1999), we expect a project with more "eyeballs" to have higher software development activity. Therefore, we encouraged the students to select projects with three or more developers.
- **Development status** (Alpha, Beta, Mature, etc). We encouraged the students to use the alpha and beta releases. These versions of the software are released to the F/OSS community for debugging and implementations of functionalities. Much project activity is centered on these versions. The mature and stable releases are not likely to generate much discussion in which students can contribute because many of the critical bugs may have been removed.
- **Programming language** (C, C++, etc). If students are to take part in coding activities, they should choose programming languages they are most comfortable with. Coding was desirable but not necessary task for this pilot study.
- **Extensive collaboration in lists/forums.** Most project activities take place in forums and lists. So it's important that students choose projects with active forums. This is mostly the case with projects that are hosted at sourceforge.net but also having their own web sites.

Each student was asked to prepare a report on his/her selected project for class presentation. In their presentations each student gave a brief history of his project and listed the project's characteristics based on the F/OSS projects selection criteria.

### 2.2 Phase 2

During this period the students learned how to register in their projects, use bug tracking systems, and browse and report bugs. Each student sent his/her project name and login details to the lecturer. These details were used to track students' activities in their projects. Every time a student submitted a bug, he/she notified the lecturer. Students were asked to continuously login to check the status of their submission. They could work in their projects anytime and anywhere they felt like. The students implemented the testing strategy shown in Figure 2. They applied testing techniques such as smoke tests, functional tests, usability tests, etc.

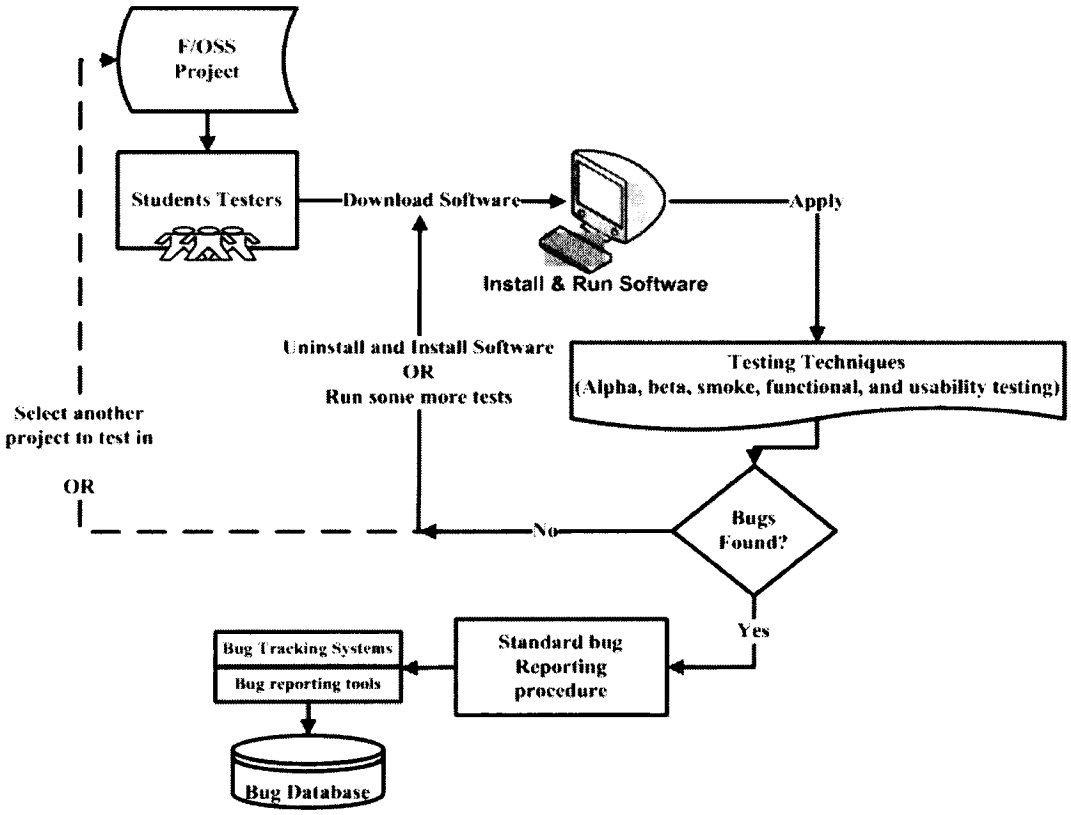


**Figure 2. F/OSS Testing Strategy.**

As shown in Figure 2, the students downloaded the software to be tested and applied various software testing techniques. This may result in the discovery of bugs (design faults, improvements to be incorporated into the next release, etc) which are then logged into the project's bug database using standard bug reporting procedure and tools (e.g. Bug Tracking Systems). Where a student was not able to find a bug, he/she ran more tests on the software or selected another project to continue testing.

In the fifth week the students were asked to make another class presentation. In presenting their experiences, the students discussed the types of bugs they found, how the bugs were found, what they thought caused the bugs, how they reported the bugs, what responses, if any, were received from other participants, and any other problems they encountered.

### 2.3 Phase 3

At the end of the pilot study the students were sent a slide presentation template and asked to make a final fifteen minute presentation. The layout of the presentation was as follows:

- Particulars (Course title, name, email, and student id).
- Project details (name, login id, website, brief history, and screen shots).
- List of Testing Activities (number of bugs found (*bfn*), bugs reported (*brp*), bugs fixed (*bfx*), and number of replies (*rep*) received). Students should give the URL of the variables *brp*, *bfx*, and *rep*.
- Like, dislike, and their future plans (if any) in the project selected.

## 3. GRADING AND EVALUATION APPROACH

At the end of the pilot study we evaluated the students based on the presentations they made in class, their participation in their respective projects, and their testing activities. Furthermore, we conducted two online surveys in order to capture the students' opinions and experiences in testing in F/OSS projects.

The presentations of the students are available at http://sweng.csd.auth.gr/~sksowe/Students%20Presentation/. The 16 projects the students tested in are shown in Table 1. The online surveys and their respective *URLs* are in Apendix 1 and 2.

The students were graded and the marks awarded as their coursework (50% of their grade) and written exams (for the other 50%). The grading was done as follows:

- **Class presentation** (10%). 3 points for each of the presentations made in Phases 1 and 2. And 4 points for the final presentation in Phase 3.
- **Project participation** (12%). Measured by the number of emails we exchanged with the student about his project
- **Working with testing tools** (13%). How a student used and understood the bug tracking system or bug database in his project.
- **Testing activity (TA) 15%.** Measured by four variables; (*bfn*), (*brp*), (*bfx*), and (*rep*).

| # | Project Name | Category | URL |
|---|---|---|---|
| 1 | Mozilla (Seamonkey) | Internet Browser | http://www.mozilla.org/projects/seamonkey/ |
| 2 | Mozilla (Firefox) | Internet Browser | http://wiki.mozilla.org/Firefox:1.5.0.2:Test_Plan |
| 3 | Imagein | Image Processing | http://sourceforge.net/projects/imagein |
| 4 | Vdrift | Games | http://vdrift.net/ |
| 5 | Cube | Games | http://sourceforge.net/projects/cube |
| 6 | Eclipse | Open Platform | http://www.eclipse.org/ |
| 7 | FloAts Mobile Agent | Mobiles & Networks | http://fma.sourceforge.net/index2.htm |
| 8 | Torcs | Games | http://torcs.sourceforge.net/ |
| 9 | Audacity | Entertainment | http://audacity.sourceforge.net/ |
| 10 | Mednafen | Games | http://mednafen.com/ |
| 11 | Stellarium | Astronomy | https://sourceforge.net/projects/stellarium |
| 12 | Dr.DivX | Playback | http://sourceforge.net/projects/drdivx/ |
| 13 | Mill3d | Games | http://sourceforge.net/projects/mill3d |
| 14 | Stunts3D | Games | http://sourceforge.net/projects/stunts3d |
| 15 | Mega Mario | Games | http://sourceforge.net/projects/mmario |
| 16 | Gloster | Games | http://gloster.sourceforge.net |

**Table 5. Projects Students Tested In**

We will now discuss the results of the students testing activities and their responses to the questions in our surveys.

### 3.1 Students Testing Activities

Results of students testing activities are described in terms of the four variables (*bfn*, *brp*, *bfx*, and *rep*). Table 2 shows a simple summary of students testing activities.

| | | *bfn* | *brp* | *bfx* | *rep* |
|---|---|---|---|---|---|
| N | Valid | 13 | 13 | 13 | 13 |
| | Missing | 0 | 0 | 0 | 0 |
| | Mean | 5.538 | 5.231 | 1.150 | 3.310 |
| | Median | 4.000 | 4.000 | 1.000 | 3.000 |
| | Std. Deviation | 3.017 | 2.743 | 1.281 | 2.175 |
| | Range | 8.0 | 9.0 | 3 | 7 |
| | Minimum | 3.0 | 2.0 | 0 | 1 |
| | Maximum | 11.0 | 11.0 | 3 | 8 |
| | Sum | 72.0 | 68.0 | 15 | 43 |

**Table 2. Descriptive Statistics of Students Testing Activities**

In total, the 13 students tested in 16 F/OSS projects, found 72 bugs, reported 68, fixed 15, and received 43 replies from the F/OSS communities in their projects. The mean values of bugs found and reported per student were 5.54 and 5.23, respectively. These figures show that the students reported slightly less bugs than they found, because some of the bugs they found were already reported. Even though the students performed well in finding and reporting bugs in their projects, they did not do well in fixing bugs (mean=1.15). This is because they were not required to do any coding in this part of their course.

As shown in Figure 3, students #2, #3, #4, #5, #7, #011 and #13 have low to moderate performance in all of their activities. Students #1, #6, #8, #9, #10, and #12 have performance only in three variables (bfn, brp, rep), with student #6 performing much better than the rest.

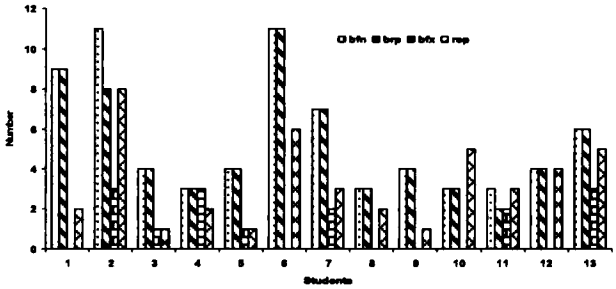


**Figure 3. Stacked Bar charts showing how the students perform in each variable.**

### 3.2 Students Opinions and Experiences: Surveys

We invited the students via email to complete two anonymous online surveys. The surveys were designed and published on the department's website using PHPSurveyor. The surveys items were meant to measure

- the pedagogical value of the study,
- students' opinions and experiences about software testing in F/OSS projects, and
- how much the students have benefited by contributing and learning from testing in F/OSS projects.

Survey one was conducted in week 6 and consisted of 21 items. We called it an *intervention survey* because it allowed us to intervene early and focus attention on difficulties students were having (e.g. ease of finding a project, process of reporting bugs). Survey two was conducted in week 13 and consisted of 19 items. The response rate for both surveys was 84.62% (11 out of 13 students). Seven items (Table 3) from survey one were repeated in survey two so that we could compare students' responses to the common questions and see how their motivation and perception has changed overtime.

Comparing the responses of the students to the common questions, we were able to make the following conclusions:

**ST1Q1 vs ST2Q3:** All 11 students responded "Yes" in both surveys. This means that throughout the pilot study students enjoyed software testing in their projects.

| | *Variables* | |
|---|---|---|
| Items/Questions | *Survey 1* | *Survey 2* |
| Do you enjoy software testing in F/OSS projects? | Q1 | Q3 |
| Did you find it easy to get a project to participate in? | Q2 | Q18 |
| Was it easy to find bugs in the software in your project? | Q3 | Q5 |
| Was the process of reporting bugs easy? | Q4 | Q6 |
| Would you prefer to do other courses in Open Source Software? | Q9 | Q8 |
| Did you read and understand the bugs others reported? | Q10 | Q11 |
| Are you considering participating in you project after you graduate? | Q20 | Q14 |

**Table 3. Questions common to both surveys**

As shown in figure 4, in both surveys the students expressed that it was not easy to find a project to participate in. This was surprising because we were expecting the students to find projects very easily within the myriad of F/OSS projects at sourceforge.net.

In both surveys the answers are the same (Figure 5). Most of the students found it difficult to find bugs in their software. This was the case at the beginning. But as they became familiar with the software, the rate of finding bugs increased gradually.

Even though finding bugs was difficult, Figure 6 shows that most of the students reported the bugs they found with relative ease. However, this was also easy at the beginning but became gradually difficult as they could not find new bugs.

Students' response to this item in both surveys, Figure 7, shows that most of them would prefer to have their other courses taught using F/OSS methodology, especially towards the end of the study. In survey two all the students answered in the affirmative.

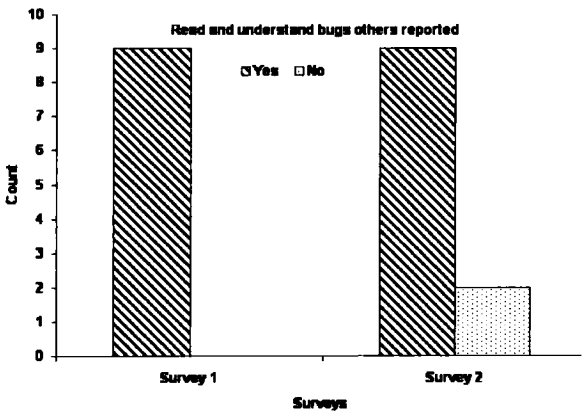


**Figure 4. ST1Q2 vs ST2Q18**

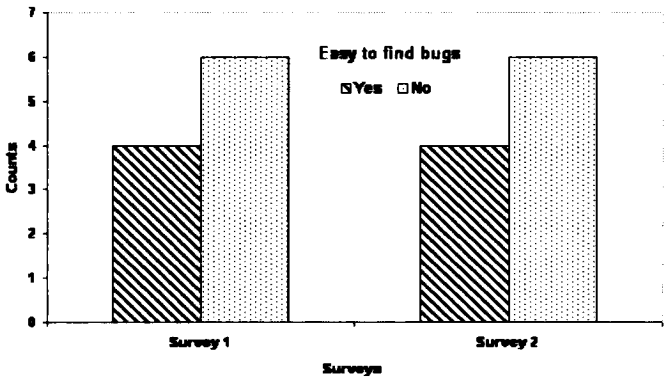


**Figure 5. ST1Q3 vs ST2Q5:**

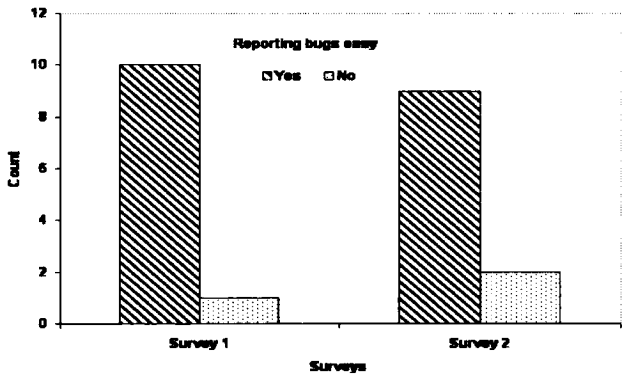


**Figure 6. ST1Q4 vs ST2Q6:**

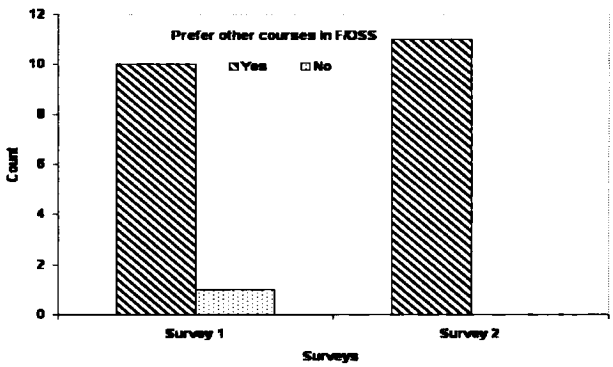


**Figure 7. ST1Q9 vs ST2Q8**

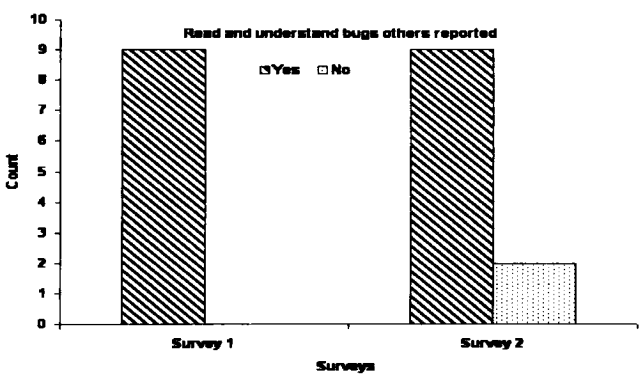


**Figure 8. ST1Q10 vs ST2Q11**

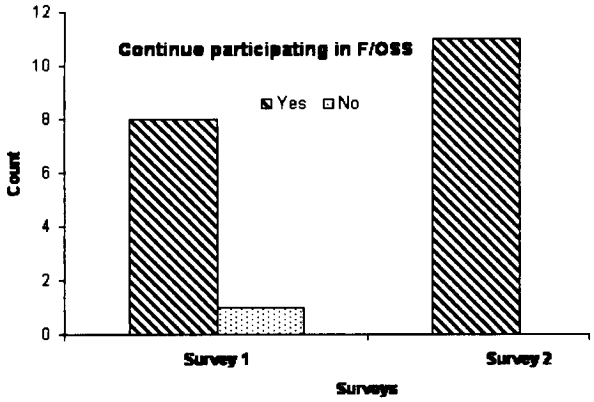


**Figure 9. ST1Q20 vs ST2Q14:**

As shown in Figure 8, the students expressed that they read and understood bugs others reported in their projects. Two students did not answer this question in survey one but in survey two those students expressed that they did not understand some of the bugs reported in their project.

The discrepancy in the two surveys, as shown in Figure9, show that two students did not answer this question in survey one. However, towards the end of the study in survey two, all the students were considering participating in their projects after they graduate.

For the rest of the items (Figure 10), we grouped the responses of the students into four categories.

**3.2.1 The teaching and learning context:** The students acknowledged the assistance we gave them in selecting their projects (N=9). This took the form of prompt replies to their email queries, emailing them handouts, and informing them of any developments we noticed in their projects. We only intervened in their testing activities when it was absolutely necessary. For example, a student might decide to discuss a bug before reporting it (N=5). Thus, more than half (N=7) of the students reported getting assistance from the lecturer in their testing activity.

Furthermore, the students seems to be satisfied with the mode of communication (N=11) - via email. However, 8 out of 11 students reported that they would have preferred more face-to-face communication with the lecturer. In survey 2, all the students (N=11) reported that they worked with another student in their project, because they preferred this mode of working (N=11).

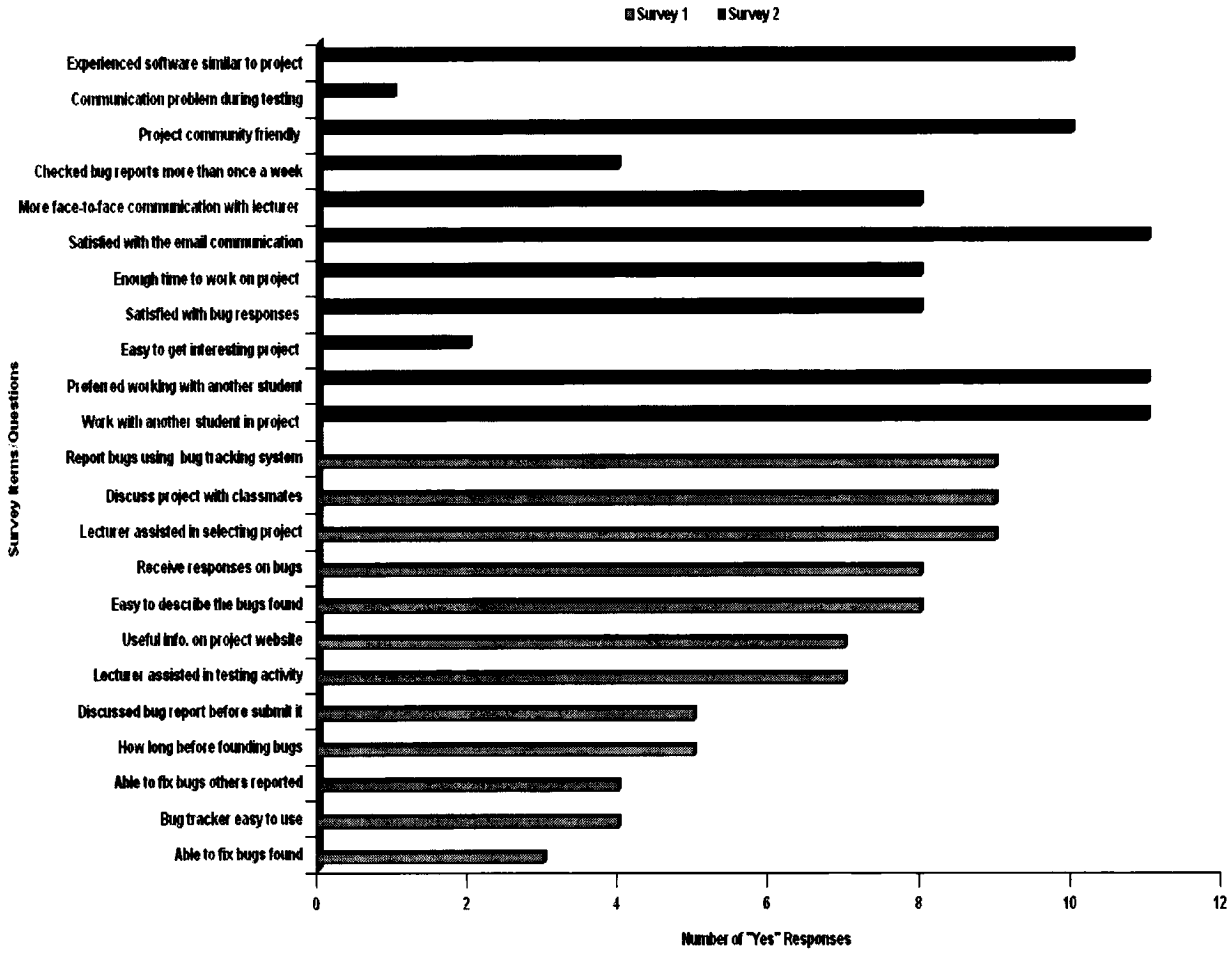


**Figure 10. Survey Responses**

**3.2.2 Students' involvement in F/OSS projects**: In both surveys the students expressed that it was not easy to find a project to participate in. Even a higher number (N=9) in survey 2 reported that it was not easy to get a project which interested them. Two students felt otherwise. However, ten students managed to find projects in which they have experienced similar software. This helps to explain why most of the students chose projects in the games category (see Table 1). The students reported that it took them, on average, 1-2 days before they could find any bugs in the software in their projects. Most of them logged-in to check the status of their bug reports at least once a week. Others did so at various times. Eight out of eleven students reported that they had enough time to work on their projects. Below is a summary of some of the comments students made when asked to list three problems they encountered when selecting a project:

- *There is a variety of projects to choose from.*
- *Some programs also required me to install other programs before I could run them.*
- *Many projects were 'dead' or inactive.*
- *Not so many active and beta testing projects available that would fill my needs.*
- *Most of the projects I tried did not address the every day needs for me as a student.*

**3.2.3 Students testing experiences**: Most of the students (N=9) preferred using the bug tracking systems (BTS) in their projects to report bugs. However, only nine students reported that BTS are easy to use. In survey one, eight students reported that it is easy to describe the bugs they found. However, only three students were able to fix the bugs they found and only four were able to fix bugs others reported in their projects. Below is a summary of some of the comments students made when asked to list three problems they encountered when submitting bug reports:

- Some of the projects am interested in did not support bug tracking system.
- Others had found the same bugs before me.
- Some bug reporting document guidelines are too long and has many versions.
- I couldn't describe exactly some bugs.

**3.2.4 F/OSS community response**: Towards the end of the program, ten out of eleven students reported that their projects' communities are very friendly and responsive. Eight students reported that they received responses to their bug reports and that they are satisfied with the responses they received. Furthermore, seven students (including all the three students who tested in Mozilla) reported that their projects provided useful information to help them in their testing activities.

## 4. EXPERIENCES AND LESSONS LEARNED

Below is a list of our experiences, lessons learned and some practical problems we encountered in the implementation of the F/OSS framework. After our encounter with the students in Phase 1, our only source of contact with them was through email exchanges (135 emails in 87 days). We had only three meetings with all the students. These were during the class presentations, which also acted as brainstorming sessions.

### 4.1 The Positives

We believed that this program has provided many benefits to the students:

1) Practical experience with the way software testing (especially smoke and usability testing) is done and how large and complex F/OSS projects work.
2) Experience in working in virtual distributed software development environment.
3) Experience in writing good bug reports and communicating their ideas clearly to 'colleagues' they have not met face-to-face.
4) Opportunity to work with essential software testing and bug reporting tools they might use when they graduate.
5) Students were easily integrated in their project teams. It is questionable whether this would be the case with non-experienced students working with professionals where many problems might arise (intimidation of students, high abandon rate, etc.).

**4.1.1 Schedules**: The students have different courses and varying schedules. The flexibility of the F/OSS framework in terms of when and where to test gave them enough time to devote to other courses and interests.

**4.1.2 Learning opportunities**: We learned from the students as well. For example, as a result of two students testing in a project using the Mantis Bug Tracker, both the lecturer and the adviser were exposed and had to learn this tool. The students benefited by having opportunity to have practical experience with software they read about in their textbooks.

### 4.2 The Negatives

4.2.1 Specialization: The students worked on only one aspect of the ISE course (software testing). While we are comfortable that they experienced testing in F/OSS projects, we are concerned that this may have prevented them from experiencing other aspects of software development (documentation, design, coding).

**4.2.2 Collaboration**: The students were encouraged but not required to work in groups/pairs. We had only one instance of collaboration among the students (two worked in one project). But in the surveys the students expressed that they indeed collaborated and discussed with their classmates.

**4.2.3 Evaluation problem**: We had some "overachievers" who knew Linux and UNIX well. They also did well in finding and reporting bugs and helped other students. We had no way of rewarding these extra-curricular activities.

**4.2.4 Sample size**: Out of the 150 students in the ISE course, only 15 volunteered at the beginning. A larger number of volunteers would have enabled us to make more sound generalizations about our results. Retrospectively, the small number allowed us to effectively interact with the students and made it possible for each student to present his/her work. As Carrington and Kim (2003) found out this might be impossible to achieve with a larger group.

**4.3 Possible improvements**

**4.3.1 Synchronous communication**: We might set up a Web-based discussion forum where we could 'meet' and discuss with the students at least twice a week. But we feared that this would impose some restrictions on the students because they would then be required to be online at a scheduled date and time. Using Instant Messaging (IM) to communicate with the students was another available option. Either would have been appropriate and much easier than composing and sending emails. However, communication with the students would have been difficult to archive, search and monitor.

**4.3.2 Encroach on students' freedom**: We did not intervene when the students were selecting their projects, resulting in most of them selecting projects in the games category from the sourceforge.net portal. We believe that this provide problems for diversification. We speculate that if we had pre-selected and presented a diversity of projects to the students, we would have had a better variety of projects. But this would also contradict the F/OSS norm of freedom to choose what you want to work on. As Raymond (1999) noted and affirmed by Hippel and Krogh (2003), one of the norms clearly expressed in the F/OSS community is that work cannot be mandated and enforced.

Furthermore, responses to both surveys indicate that two students did not complete the surveys. We could have gained 100% response rate if we had made both surveys non-anonymous and therefore mandatory.

**4.3.3 Group work/Community formation**: At the beginning we considered organizing the students into groups (3-4 students). Then each group would test in one project (e.g. Mozilla Firefox or Mozilla Thunderbird). In each group we could then have a task-leader (akin to a project leader) selected based on his degree of involvement and contribution to the group's task.

**4.3.4 Grading**: In assessing the students we wanted to use the number of views a bug report received from the project's community as a fifth variable (in addition to *bfn*, *brp*, *bfx*, *rep*). However, only in three of the projects the students selected was this available. Moreover, we suspected that these numbers may have been inflated because the students may have viewed their own submission many times. There was a total of 729 views to students contributions in 3 projects (mean= 56.1, Std. Deviation= 115.754). We posit that the number of views a contribution receives in an F/OSS project is a good indication of how interesting or uninteresting that contribution is. Paradoxically, it is very difficult or impossible to see who viewed what.

**4.3.5 Surveys**: Applying Likert type questions for some items, especially those related to students' attitudes or their opinions, could produce much more interesting information or results.

**4.4 Difficulties**

**4.4.1 Lecturer as Project Manager**: We realized in this pilot study that the lecturer's role is more than preparing and delivering 2hrs lectures. There was no fix lecture hours for this aspect of the ISE course (students work on their projects anytime they feel like). The lecturer needed to assume the role of a manager in all the sixteen projects so that he knew what is going on (tracking bug reports and responses). The role enabled the lecturer to advise the students better when they sent their email queries. Taking on all of the responsibilities on top of the usual teaching schedule was a gratifying burden. Recruiting another person (e.g. a PhD student specializing in F/OSS) to take charge of students activities when implementing such a framework seems a good idea.

**4.4.2 When to stop scoring points**: Students motivation was very high in this program. In one class presentation session one of the students commented "*...we* (the pilot group) *are the modern students. We do the modern stuff, Linux*" (apparently he equates Linux with F/OSS). So it was difficult to tell the students "Now stop finding and reporting bugs". Until the very day we graded the students, some were reporting bugs and we had to adjust their marks accordingly. We still continue getting emails about their activities and we reply with encouraging comments.

**4.4.3 Post projects activities**: In a corridor discussion with one of the students who failed in the final exams, he commented that "*I don't care if I fail! I have been using the software before the start of the program anyway. Now I know the project and people writing the software, am still in it and will continue...*"

## 5. DISCUSSION AND VALIDITY THREATS

In this pilot study, the results of the students' testing activities show that some of them not only learnt knew skills but have experienced, for the first time, how to do software testing in F/OSS projects (*OBJ1*). They might not graduate from this pilot study as professional software testers but the study has given them a new start as some of them acknowledged in the surveys (Section 3.2.3). Some students tested solo in their projects but some collaborated and tested in groups, thus giving them experience in working as part of a software Quality Assurance team and making contributions to the F/OSS community (*OBJ2*). Besides, they experienced that working in a distributed software development environment such as in F/OSS projects not only need patience (see their comments in section 3.2.2 and 3.2.3), but also one needs to negotiate and communicated his ideas well in order to be appreciated by the community of the project in which he/she participates (*OBJ3*). As seen in section 4.1, perhaps the most positive aspect of this exercise for both the students and lectures was the opportunity to learn to use new software testing tools (*OBJ4*).

While such an pilot study helps to meet some of the SE learning objectives advocated by the IEEE/ACM curriculum guidelines, implementing a full SE course using the benefits inherent in the F/OSS methodology is not with its challenges as discussed in Section 4.2 and 4.4.

**Validity threat:** The validity of our pilot study could be compromised by the size of our sample. 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 SE 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.

Furthermore, based on discussions about this pilot study in conferences (Sowe, *et. al.*, 2006b), meetings (Flosscom.net Kick-off meeting, November, 2006), and workshops (SQO-OSS Workshop, 2006), we learnt that there are some pending research, including:

- research to shed light on how we can blend the F/OSS teaching and learning environment with that of the formal SE teaching and learning context in colleges and universities,
- a broad understanding of the F/OSS pedagogy,
- F/OSS evaluation and assessment methodologies,
- how F/OSS can improve the quality of teaching and learning, and
- how to create a partnership between students and F/OSS developers, projects and industry.

## 6. CONCLUSION AND FUTURE WORK

This paper has described a pilot study in which students were involved in software testing in F/OSS projects. The F/OSS framework discussed emphasized a teaching and learning context in which the students were exposed to "real-world" software engineering projects. The evaluation of the framework was based on students' participation in their projects and the results of two online surveys we conducted. In section 4 lists key pedagogical issues which may be peculiar to F/OSS or this framework, but serve as important guidelines for CS lecturers and SE educators. The implementation of the framework in a formal SE course shows that it is feasible to teach project-based SE courses such as the Capstone projects advocated by the joint IEEE/ACM curriculum guidelines, using F/OSS methodology.

As our future work, we utilized our experience from this pilot study. We have an ongoing pilot study of a similar nature which will be concluded in September, 2007. We have over 58 volunteers (experimental group) and 75 non-volunteers (control group). We intend to compare the performance of the two groups. The variables we would be assessing are the following;

- Size in terms of number of developers and type of FLOSS project(s) each student chose to work with.
- Time dedicated to bug finding and bug reporting.
- There opinions of F/OSS, using modified versions of our online surveys.
- Learning style of the students
- Content analysis of emails we exchange with the students and the emails the students exchange with others in their projects. This would give valuable information regarding the type of interaction that involved the students and their project's environment (or F/OSS communities) as well as the interaction with his/her instructor.
- The role of students as mentors. We have last year's students who volunteered to work with the new students this semester.
- A survey of opinions of other CS staff about the F/OSS teaching and learning framework we discussed here.

## 7. ACKNOWLEDGEMENT

We wish to acknowledge the participation of Prof. A. Lefteris of the Department of Informatics, Aristotle University. He proposed an alternative statistical evaluation approach which we could not report in this paper. The statistical analyses are available on request.

This work was partially funded through the European Commission, DG Education and Culture, Socrates programme, Minerva action line, project ref: 229405 - CP -1-2006-1- PT - MINERVA – M (http://www.flosscom.net/).

## 8. REFERENCES

Alzamil, Z. (2005), "Towards an effective software engineering course project." Proceedings of the 27th international Conference on Software Engineering, ICSE '05. ACM Press, pp. 631-632.

Barahona, J. M., Tebb, C. and Dimitrova, V. (2005), "Transferring Libre Software development Practices to the Production of Educational Resources: the Edukalibre Project." First International Conference on Open Source Systems, Genova, Italy.

Carrington, D. and Kim, S. (2003), "Teaching Software Engineering Design with Open Source Software." 33rd ASEE/IEEE Frontiers in Education Conference, Nov. 5-8, Boulder, CO.

Edulink Project (2007). Retrieved Thursday, 10 May 2007, from http://edukalibre.org/

Faber, B. D. (2002), "Educational models and open source: resisting the proprietary university." Proceedings of the 20th Annual international Conference on Computer Documentation, SIGDOC '02, ACM Press, pp. 31-38.

FLOSSCom Project (2006). Retrieved Oct. 20, 2006, from http://www.flosscom.net/

"Flosscom.net Kick-off meeting", November, 2006http://flosscom.net/index.php?option=com_content&task=view&id=23&Itemid=40

German, M. D. (2005), "Experience teaching a graduate course in Open Source Software Engineering." Proceedings of the first International Conference on Open Source Systems. Genova, pp.326-328.

Hans, V. (2005), "Some Myths of Software Engineering Education." Proceedings of the 27th international Conference on Software Engineering, ICSE '05. ACM Press, pp. 621-622.

Hippel, E., and Krogh, G. (2003), "Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science."Organization Science, Vol. 14, 2003, pp. 209-223.

IEEE/ACM Joint Task Force on Computing Curricula, Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Retrieved November 27, 2005, from http://sites.computer.org/ccse/SE2004Volume.pdf

Liu, C. (2005), "Enriching software engineering courses with service-learning projects and the open-source approach."

Proceedings of the 27th international Conference on Software Engineering, ICSE '05. ACM Press, pp. 613-614.

Mantis Bug Tracker (2006), Retrieved April 27, 2006, from http://www.mantisbt.org/

Megias, D., Serra, J., Macau, R. (2005), "An International Master Programme in Free Software in the European Higher Education Space." Proceedings of the first International Conference on Open Source Systems. Genova, pp.349-352.

SQO-OSS Workshop, "Open Source Software Workshop" (2006), Open Source Software: Research Communities and Industries. Thessaloniki, 20 December, 2006, available at: http://www.sqo-oss.eu/events/public-workshop/

Ozel, B., Cilingir, B., Erkan, K. 2006 (Eds.) Towards Open Source Software Adoption. OSS 2006 tOSSad Workshop proceedings, Como, Italy, pp.79-88

PHP Surveyor (2006), Retrieved April 27, 2006, from http://www.phpsurveyor.org/index.php

Raymond, S. E. (1999), The Cathedral and the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly, Sebastopol, USA.

Sowe, S.K., Karoulis, A., Stamelos, I., Bleris. G.L. (2004), "Free/Open Source Software Learning Community and Web-Based Technologies". IEEE Learning Technology Newsletter, Vol. 6, No. 1, 2004, pp. 26-29.

Sowe, S .K., Stamelos, I., Deligiannis, I. (2006a), " Framework for Teaching Software Testing using F/OSS Methodology." In Damiani, E., Fitzerald, B., Scacchi, W., Scott, M., Succi, G.,(Eds.), (2006). IFIP International Federation for Information Processing, Open Source Systems, Vol. 203, (Boston: Springer), pp. 261-266

Sowe, S.K., Stamelos, I., Angelis, L. (2006b), "An Empirical Approach to Evaluate Students Participation in Free/Open Source Software Projects". IADIS International Conference on Cognition and Exploratory Learning in Digital Age CELDA 2006, 8-10 December, 2006, Barcelona, Spain, pp.304-308

Sourceforge.net (2006), Retrieved April 27, 2006, from http://sourceforge.net/

Pfleeger, L. S. 1998. Software Engineering. Theory and Practice. Prentice Hall, pp. 278-392.

Lakhani K. R, and von Hippel E. (2003), "How open source software works: "free" user-to-user assistance". Research policy, Vol. 32, No. 6, 2003, pp. 923-943.

Holtgrewe U. (2004), "Articulating the speed(s) of the Internet: The case of Open Source/Free Software. Time & Society, Vol. 13, No. 1, 2004, pp. 129-146.

Sowe, S.K., Angelis, L., Stamelos, I. (2006c), "Identifying Knowledge Brokers that Yield Software Engineering Knowledge in OSS Projects", Information and Software Technology, Vol. 48, No. 11, 2006, pp. 1025-1033.

## AUTHOR BIOGRAPHIES

**Sulayman K Sowe** is a PhD student at the department of Informatics, Aristotle University of Thessaloniki, Greece. He received a Bed. in science education from University of Bristol, UK (1991) and Advance Diploma and MSc. in computer science from Sichuan University, China (1997). Mr. Sowe also holds a Higher Teachers Certi-ficate (HTC) form The Gambia College, Brikama Campus. He was a part-time lecturer in Information Technology at the University of The Gambia (2002). He worked at the Department of State for Education, The Gambia as the director of Information Technology and Human Resource Developmen, as a System Administrator and Assistant Registrar II for the West African Examinations Council, and as a Database Manager for the Medical Research Council. His research interests include Free/Open Source Software Development, Knowledge Sharing, Software Engineering Education, Information Systems Evaluation, and Social and Collaborative Networks. He is also working on several projects related to Free/Open Source Software financed by Greece and the European Commission Information Society Technologies (IST) Programmes. He is the co-editor of the book "Emerging Free and Open Source Software Practices", Idea Group Inc., 2007.

**Ioannis G. Stamelos** is Assistant Professor at the Aristotle University of Thessaloniki, Dept. of Informatics and Teaching Consultant at the Hellenic Open University. He received a degree in Electrical Engineering from the Polytechnic School of Thessaloniki (1983) and the Ph. D. degree in computer science from the Aristotle University of Thessaloniki (1988). His research interests include empirical software evaluation and management, software education, agile methods and open source software engineering. In particular he has researched various aspects of open source projects, knowledge sharing in open source communities and the efficiency of ODL tools. He is author of approx. 70 scientific papers and has co-edited 3 books.

**Appendix 1.**

1. **Students PowerPoint Presentations** (http://sweng.csd.auth.gr/~sksowe/Students%20Presentation/)

2. **Students Testing in F/OSS Projects: Survey I** (http://swserv1.csd.auth.gr/survey/index.php?sid=15)

ST1Q1: Do you enjoy taking part in software testing in F/OSS projects? Yes No
ST1Q2: Did you find it easy to get a project to participate in? Yes No
ST1Q3: Was it easy to find bugs in the software in your project? Yes No
ST1Q4: Was the process of reporting bugs easy? Yes No
ST1Q5: Did you get any responses from the forums you reported your bugs? Yes No
ST1Q6: If your project is hosted at sourceforge.net was the bug tracker easy to use? Yes No
ST1Q7: Was it easy to describe the bugs you found? Yes No
ST1Q8: Did you find any information on the website of your project which helped you understand how to report bugs? Yes No
ST1Q9: Would you have preferred to do other courses using Open Source Software? Yes No
ST1Q10: Did you understand the bugs others reported in your project? Yes No
ST1Q11: Are you able to fix any of the bugs you found? Yes No
ST1Q12: Are you able to fix any of the bugs others reported? Yes No
ST1Q13: Did you get much assistance from the lecturer to help you selecting a project? Yes No
ST1Q14: Did you get much assistance from the lecturer to help you in your testing activity? Yes No
ST1Q15: List 3 problems you encountered when selecting a project? Please write your answer here:----------
ST1Q16: List 3 problems you encountered when submitting your bug reports? Please write your answer here:----------
ST1Q17: How long have you used the software before you found any bugs?
Please choose *only one* of the following: 1-2 days 3-5 days 1 week
ST1Q18: Do you prefer discussing your bug report with the lecturer before you submit it to the forum? Yes No
ST1Q19: How often do you discuss you project with others? Please choose *all* that apply
I never discuss my project activities I sometimes discuss my project with my classmates
I sometimes discuss my project with the lecturer
ST1Q20: Are you considering participating in the project you selected after graduate? Yes No
ST1Q21: Where did you report bugs? * Please choose *all* that apply:
Using the Bug Tracking System Tech Support or Future Requests Tracking System Using Public Forums

3.. **Students Testing in F/OSS Projects: Survey 2** (http://swserv1.csd.auth.gr/survey/index.php?sid=18)

ST2Q1: Did you work with another student in your project? Yes No
ST2Q2: Would you have preferred to work with another student in your project? Yes No
ST2Q3: Did you enjoy participating in software testing in Open Source Software Projects? Yes No
ST2Q4: Was it easy to get a project which interests you? Yes No
ST2Q5: Was it easy to find bugs in the software in your project? Yes No
ST2Q6: Was it easy to report your bugs? Yes No
ST2Q7: Are you satisfied with the answers you got about your bugs? Yes No
ST2Q8: Would you like to do other courses in Open Source Software? Yes No
ST2Q9: Did you have enough time to work in your project? Yes No
[Only answer this question if you answered 'No' to question 'ST2Q9 ']
ST2Q10: How much more time do you need to work on your project?
Please choose *only one* of the following: 1 more week More than 1 week More than 1 week
ST2Q11: Did you read bugs others reported in your projects? Yes No
ST2Q12: Are you satisfied with the communication you had with Dr. Sowe through emails? Yes No
ST2Q13: Would you have preferred more face-to-face meeting with [the lecturers] to discuss your project? Yes No
ST2Q14: Are you going to continue participating in your project? Yes No
ST2Q15: How often do you login to check your bug reports? Please choose *all* that apply
Everyday Every 2 days Once a week Mostly during the weekends
Sometimes at Internet Cafes Mostly at home When am at the University At a friend's place
ST2Q16: Was the people in your project friendly? Yes No
ST2Q17: Did you have any communication problem during your testing? Yes No
ST2Q18: Did you find it easy to get a project to participate in? Yes No
ST2Q19: Did you have experience with software similar to the one in your project? Yes No