STUDENTS' PERSPECTIVES ON LEARNING SOFTWARE ENGINEERING WITH OPEN SOURCE PROJECTS

Lessons Learnt after Three Years of Program Operation

Pantelis M. Papadopoulos[1], Ioannis G. Stamelos[1] and Andreas Meiszner[2]
[1]*Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece*
[2]*United Nations University-Merit, Keizer-Karelplein 19, Maastricht, The Netherlands*

Keywords: Software Engineering Education, Open Education, Authentic Learning, Open Source, Online Learning, Learning Community, Project-based Learning.

Abstract: This paper presents the results after three years of running of an instructional method that utilizes free/libre open source software (FLOSS) projects as tools for teaching software engineering in formal education. In the last three academic years, a total of 268 juniors majoring in Informatics (in a 4-year program) participated in study, assuming the roles of testers, developers, and requirements engineers. Students appreciated the benefits gained by the method and identified aspects that require further improvement. In the following, we present (a) the details of our method, (b) students' opinions as recorded through a questionnaire including both closed and open ended questions, and (c) conclusions on how the use of FLOSS projects can be applied and proved beneficial for the students.

1 INTRODUCTION

The domain of software engineering poses a considerable complexity for the students and may be hard to teach. Learning within the domain relies largely on theoretical concepts and models and their application in ill-structure contexts. In other words, software engineering is anchored to the real world and students' involvement in a project is a very popular method of instruction. However, where most of the instructional approaches engage students in fictitious projects in the safe environment of the class, we propose the use of real free/libre open source software (FLOSS) projects as educational tools.

FLOSS projects are based on open, self organized communities of volunteers, that manage to support software development, support and maintenance in an unprecedented way. This unique kind of virtual community provides an excellent environment for learning how to communicate with, cooperate with and ultimately learn from other members of the community (Stamelos, 2008).

There are three main approaches for using FLOSS projects in formal education (Meiszner et al., 2009):

1. The 'inside approach' refers to the practice of taking the principles found in FLOSS communities and applying them within the higher education context. In line with Fischer's work (2007), this approach involves mapping the key principles onto education and includes an evolutionary growth of the course and its environment. Within the 'inside approach' institutions might also decide to 'open up' their virtual learning environments to fellow universities or the general public to view what is going on within the environment. This scenario might be relatively moderate to implement since the technology should be already in place at most higher education institutions, although admittedly modifications very likely would be necessary.
2. The 'outside approach' at which institutions would send out their students into already well established and mature environments to engage and collaborate within those communities on pre-defined tasks. In contrast to the inside approach, the outside approach might take traditional education as the starting point by providing theoretical information and then send the students "outside" to find well established communities, such as the FLOSS ones, to work within those communities and to apply and deepen their theoretical knowledge.

In particular for the area of software engineering, this approach might be suitable due to the existence

M. Papadopoulos P., G. Stamelos I. and Meiszner A..
STUDENTS' PERSPECTIVES ON LEARNING SOFTWARE ENGINEERING WITH OPEN SOURCE PROJECTS - Lessons Learnt after Three Years of Program Operation.
DOI: 10.5220/0003922803130322
In *Proceedings of the 4th International Conference on Computer Supported Education* (CSEDU-2012), pages 313-322
ISBN: 978-989-8565-07-5

of a large number of mature FLOSS projects and a myriad of educational resources. The outside approach might be the least complex and almost cost neutral; and therefore relatively easy to implement.

3. If we view the inside and the outside approaches as opposite ends of a spectrum, then there is clearly a range of blended, hybrid approaches in the middle, which take components of both elements. The drawback of the hybrid approach might be that it probably requires the most drastic overhaul of higher educational practices and might be the most complex to implement.

There are already a number of studies exploring the potential of FLOSS projects in education. Jaccheri and Osterlie (2007) report on a course at the Norwegian University of Science and Technology that is based on the involvement of students in the NetBeans project and their interaction with its community. At the Athens University of Economics and Business (Greece), in the context of a master level course titled "Advanced Topics in Software Engineering", students are asked to participate and produce code in FLOSS projects (Spinellis, 2006). Staring et al. (2005, 2006) also claim that "involving students in large scale, international open source projects has a potential for transformation of the relationship between students, educational institutions and society at large". Lundell et al. (2007) report their experience from a practical assignment "designed to give students on an Open Source Masters course an insight into real involvement in Open Source projects" at the University of Skövde (Sweden). They also report on a reduced exercise for undergraduate students related to FLOSS. The authors found out that "the learning experience was both positive and valuable in that it gave real insight into Open Source participation". They also report that students were further encouraged to keep on participating in Open Source projects even after their course was completed.

Surprisingly, the underlying technology used by most FLOSS projects is relatively simple, yet mature, usually including versioning systems, mailing lists, chats, forums, wikis or similar knowledge bases. Additionally, free web based services such as Sourceforge provide each FLOSS project with an initial working and community environment therefore facilitating the take off of new projects (Meiszner, 2007).

From a pedagogical perspective learning in FLOSS is characterized by self-studying, project-based learning, problem-based learning, inquiry-based learning, collaborative learning, reflective practice or social learning. It is not assumed that those pedagogies were deliberately set out, but rather that due to the structure, approach and governance of FLOSS communities certain pedagogies have emerged (Glott et al., 2007).

In the next section, we present our instructional method along with the three roles our students are able to take as members of FLOSS projects, the learning environment we used to support the activity, and the assessment method. In section 3, we present students' responses regarding various characteristics of the activity. Students refer to the strengths of the approach and underline aspects that need further improvement. Finally, we provide discussion on the results and concluding remarks.

2 THE INSTRUCTIONAL METHOD

We first started using FLOSS projects as a medium of instruction in software engineering during the academic year 2005/2006. The way our students were engaged in the open source community went through changes over the next years. Since the academic year 2008/2009, the instructional method remains the same. This section presents the specifics of the method we have applied in our university in the last 3 years.

2.1 Participants

The core course "Introduction in Software Engineering" (ISE) is typically offered in the 5th semester in our Informatics program (in a four years study program). The duration of the course is 12.5 weeks and approximately 150 students enroll in the course each semester. Since this is a core course, a passing grade is necessary for all the students, in order to graduate. A falling grade means that the student will have to repeat the course during the next academic year. This causes the actual number of enrolled students to fluctuate over the years.

The FLOSS activity is introduced as a course assignment to the students. The assignment counts for 40% of the total course grade and it was optional the first two years with 31 (2008/2009) and 50 (2009/2010) students participating respectively, and mandatory the last year with 187 (2010/2011) students completing the assignment.

2.2 Procedure

In the course, we apply a hybrid approach combining formal, distance, exploratory and project-based learning. Students receive printed material (books, tutorials, papers) and online resources and they learn about the theoretical concepts of the domain through a series of in-class lectures. The typical form of instruction continues, until the students are ready to move from the book examples to real cases. At this point selected FLOSS projects are presented in class and students have the opportunity to see how theoretical principles are applied in real world contexts.

In the next step, we introduce the assignment to the class. Students have to search and select one of the FLOSS projects readily available online, and actively participate in the project, choosing one of these roles: (a) tester, (b) developer, or (c) requirements engineer. We chose these roles to give the opportunity to students to approach FLOSS projects from three different points of view. Each role has specific needs and immerses the students in the software engineering world. The ability to choose the role that fits better their characteristics is very important to the students. While some may have a knack for finding bugs, others may prefer the analytical work behind a requirements document, or – if they have confidence in their programming skills – the challenge of developing something new and extending the functionality of a FLOSS project.

At this point, a supporting learning environment becomes available for the students (more on this on section 2.2.4) and detailed guidelines and instructions are given for each of the three roles.

2.2.1 The Role of the Tester

Students opting for the tester's role have to test projects and find a small number (5-10) of non-submitted bugs, report these bugs properly in the bug tracking system (BTS) of the project (or in a developers' forum, if such a system is not in place), and – optionally – assist the development team address these bugs. Testers have to follow a 5-step procedure:

STEP 1: Find a project. Students can search for a FLOSS project anywhere they want, although a list of webpages with a big volume of projects is proposed to them (e.g., sourceforge.net, tigris.org, etc.). Students should browse the available categories of projects (e.g., finance, games, enterprise, etc.) and find something they like. To help them in their selection, we propose a list of criteria. First of all, the project has to be compatible with the operation system the students use at home. Then, they should check the maturity level of the project. Stable/mature projects may have fewer bugs, thus making the task of finding bugs hard. Instead, the students should focus on projects in beta development status. Finally, they should check the activity of the project. A big number of developers, the existence of a busy mailing list/forum, and an up-to-date feed on the main page of the project, indicates a high activity and a bigger probability of a timely feedback for the student.

STEP 2: Register the project. After selecting a project, the student has to declare it in the learning environment used in the ISE course. Only after granting permission from the instructor, the student is able to start working on the project. For example, it is not allowed for two students to work on the same FLOSS project with the same role.

STEP 3: Install and run the project. In this step, the student has to start using the project to understand its functionalities and check the appropriate way of reporting bugs. Most of the projects use BTS, while others use a dedicated sub-forum, or a mailing list.

STEP 4: Find bugs. During the in-class lectures, students learn various techniques to test a program and find bugs (e.g., smoke, recovery, exploratory, functional, usability, etc.). They can apply multiple techniques, in order to find bugs. Next, they should check if a bug has already been reported and if not, they can submit it through the proper channel, along with sufficient information so that it could be possible for the developers to reproduce it. Students should also monitor the bugs the report, in case additional information is requested.

STEP 5: Present the project. At the end, the students have to present their work to the rest of the class by submitting a detailed report to the learning environment and doing a slides presentation in class. The report should include information about the selected project, the interaction between the student and the project community, and links to bugs submissions.

2.2.2 The Role of the Developer

Students opting for the developer's role have to check the project page for desired functionalities and unsolved bugs, write appropriate code, and submit it for approval to the project team. The 5 steps a developer has to follow are:

STEP 1: Find a project. This step is similar as before. The students are able to search online the FLOSS project that better fits their needs. In

addition to the previous criteria concerning the level of maturity and activity of the project, the students should be confident and familiar with the programming language of the project. Since they have to produce functionalities and commit their source code gradually (to give time to other to comment on their work), time scheduling is an important factor in the selection process. We advice our students to also check out the "Help Wanted" section of webpages hosting FLOSS, since there are many projects looking for developers. The goal is to choose a project in which the students will receive timely feedback on their work.

STEP 2: Register the project. Same as before; the students have to declare their projects in the learning environment and get approval from the instructor to proceed with the assignment.

STEP 3: Contact the developers. The students have to contact the developers and inform them of their intention to develop parts of the source code. If the developers agree, the next thing to do is to start understanding the existing code. Otherwise, it is better to go back to STEP 1 and select another project. The support of the open source community is vital for students regardless of the project or the role they choose. When the students are ready they propose functionalities they want to implement, basing their proposals on desired features and open bugs reported in the official page of the project. If the developers agree with the proposals, they grant students access to the concurrent versions system (CVS) of the project.

STEP 4: Write source code. Finally, students can start working on the source code. They should submit their code gradually back to the project and give time to the project community to respond with corrections and suggestions.

STEP 5: Present the project. After the completion of the source code, the students should present their work, including information about the project, their interaction with the project team, and the functionalities they coded.

2.2.3 The Role of the Requirements Engineer

Students have to prepare a system requirements specification (SRS) document for a project that does not have one, is partially specified, or outdated, and submit it to the community for evaluation. The students could also propose improvements and amendments to the existing requirements as appropriate. Students acting as requirements engineers have to follow this 5-step procedure:

STEP 1: Find a project. As always, the first thing the students have to do is to find a suitable project to work on. In contrast to the advice we give to students acting as tester, students opting for the requirements engineer role are advised to find stable/mature projects, preferably without an SRS document.

STEP 2: Register the project. Once again, the students have to get approval from the instructor on the selected project to move to the next step.

STEP 3: Contact the developers. The students have to contact the developers and ask them if a SRS document is needed in the project. If such a need exists, the developers should provide the students with all the necessary information to write the document.

STEP 4: Write the SRS document. Before they start writing the SRS document, students have to spend a significant amount of time using the project to fully understand its functionalities. Only then, they are ready to start writing. The students have to produce a formal document. Hence they have to follow any available template for such a document in the webpage of the project. If such a template does not exist, the students can use one of the templates we provide for them in the learning environment. Along with the text, students have to include screenshots and diagrams, where this is necessary. The finished document should be submitted to the project community. Comments and reviews from the community should be taken into account, and – if necessary – revisions may be done.

STEP 5: Present the project. The assignment is completed when the students present their work, including information about the project, their collaboration with the project team, and the document they produced.

2.2.4 The Learning Environment

In the first two years of the study, we used the NetGeners environment, while in the third year, we transferred to openSE. Both environments were products of European projects on open education and software engineering education, in which we have been partners. While there were several tools in each environment, the subset we used for the ISE course was identical in both environments. This allowed us to apply the exact same instructional method throughout the years. NetGeners was shut down at the end of the second year, while openSE was taking over. During migration, all the material (e.g., reports, forum posts, documents, etc.) were transferred without any loss. At this point, we need

to underline that the instructional method is not grounded to the characteristics of the specific learning environment. On the contrary, the interested instructor could apply our method of using FLOSS projects to teach software engineering with a different set of tools.

The main function of the environment was to support students during the assignment. A digital library contained all the necessary resources (e.g., documents, instructions, tutorials, templates, external links, etc.). This library was updated when necessary (e.g., instruction clarification, new SRS templates, etc.).

A second very important function of the environment was to act as a hub among students, past and current. A basic set of communication tools were available (forum, blog, chat). Through them students could communicate with each other. Although each student had to work independently on a different project, the feedback from peers, assistants, and instructors was important for them. Among the communication tools, the forum was the prominent one, since it was the most organized and was holding the main volume of knowledge (e.g., FAQ section, roles sub-forums, past experiences, managerial issues section, etc.). Students also had personal blogs where they could upload information about their projects. Blogs were used mostly as journals, informing others (and most importantly the instructor) on their progress and the difficulties they had in each step. Finally, the chat was used rarely (mostly around deadlines), since students preferred other ways of synchronous communication (e.g., MSN, Skype, etc.).

After the first year, the environment served one more purpose; students were able to see what previous students had to deal with, by reading their reports and blogs. This helped students a lot, especially in selecting appropriate projects for them. Furthermore, students had a better image of what to expect. For example, how long it takes to receive feedback from others, when the right time is to abandon a project with low activity and start working on another one, what type of project is more appropriate for a specific role, etc.

Another function that was offered, but was used randomly by the students, was the ability to review each other's work. For this, a simple review form with one textbox was available in each project report and students were able to freely comment on the project. In addition, there was a 5-star rating scale for the project, much like the rating scale used in commercial sites (e.g., Amazon). Since reviewing one another was not an assignment requirement, students refrained from doing so, possibly in order to avoid conflict.

Finally, it is worth mentioning that the learning environment was open to students and instructors of other higher education institutes (HEI), and to any other interested individual. This means that it was possible for our students to receive feedback from people outside the course. Of course, most of these messages came from our past students that kept visiting the learning environment, acting as consultants, and giving advice to current students.

2.2.5 Assessment Method

An obvious parameter of student assessment was the quality of the work they produce (number and significance of bugs reported, complexity and effectiveness of code developed, clarity and usefulness of SRS document).

However, the main goal of the assignment was to immerse the students into the real world of software engineering. Because of this, the actual involvement of the student in the open source community was equally important. This is the reason why the students had to elaborate in their reports about the collaboration they had with the project team. The volume and quality of collaboration could be estimated according to the number and importance of messages exchanged in forums, mailing lists, or project pages. We encouraged our students to produce high quality work and we decided to award these efforts: if the work of a student was adopted by the project community (progress on the reported bugs, use of developed functionalities in new versions, post of SRS document on the project page), the full grade for the assignment was awarded by default. Of course, the student had still to work on the final report and present the project to the class.

Since students' success was based – up to a certain point – on the level of project activity, we allowed students to work on their assignments beyond the 12.5 weeks of the official lecturing period and submit it at a later time at 3 pre-defined dates per year – by the end of the course in February, or alternatively in June or September.

After the completion of the assignment, the students had to answer a questionnaire focusing on how they perceived the activity and what is their opinion regarding the strengths and weaknesses of our approach. The results of this questionnaire for the three-year period of this instructional method are presented in the next section.

3 STUDENTS' RESPONSES ON THE METHOD

After the end of the assignment each year, we asked students to complete online a comprehensive questionnaire, covering many aspects of the activity. We compiled the questionnaire using both closed and open-ended items and students had also the opportunity to freely submit their comments. The set of questions used addressed every aspect of the activity for all the three student roles.

The data presented here depict students' opinions over the last three years of the FLOSS instructional method. Despite the fact that the learning environment changed in the third year, the method – and thus the questionnaire too – remained the same.

Table 1 presents the number of students in each role for the three academic years.

Table 1: Number of students in each role.

	'08/'09	'09/'10	'10/'11	Total
Testers	12	27	50	89
Developers	0	7	20	27
Engineers	19	16	117	152
Total	31	50	187	268

During the first academic year, only 31 students volunteered to participate. None of them chose the role of the developer. Probably the fact that this was the first year we applied the method and the lack of previous experience from past students discouraged students from taking that role. Although, all three roles are demanding, the role of the developer requires high programming skills and confidence. It is obvious that this was the least favorite role among the students with group numbers constantly trailing behind the other two roles.

Regarding the testers, their numbers seem to double each year, while the engineers' group remained stable for the first two years and grew 8 times in the last one. As we mentioned earlier, during the first two academic years the participation in the assignment was optional, while in the third year participation became mandatory for all, as the assignment became part of the course. This is the reason why the students' population spiked in the third year. According to students' statements, writing an SRS document seemed less technical and closer to their set of skills. While we believe that in many cases, students underestimated their technical competencies, we have to keep in mind that this was the first time for the students that their work would be submitted for evaluation to a greater community that exceeds the safe and familiar environment of the university. The fear of receiving negative comments for their work made students choose roles and projects that would seem more feasible for them.

This fear, however, does not have a real base. The culture behind the open source community dictates that any contribution, especially since it comes free and voluntarily, should be appreciated. Thus, we maintain that it is not a matter of strengthening our students' skills, but changing their attitudes about being members of a bigger community.

Table 2 presents students' answers to some of the most important closed-type questions. In general, students in all the three groups have a very positive opinion about the activity (Q4). Some differences appear and they can be explained based on the characteristics of each role.

The first difference appears in item (Q1), where developers and engineers said that they had to spend at least two weeks using the project and understanding its functionalities before starting to write code or explain existing functions to others. In comparison, testers were able to start finding bugs on the third day. Some bug are more obvious than other and do not require deep knowledge of the project. However, after all groups start working, the degree of difficulty becomes comparable for all students (Q5, Q6).

The task of finding an appropriate project is not simple. Students made it clear that they had difficulties (Q2). It appears that it was more difficult for testers that tried on average more than 10 projects, while the others had to try on average only five (Q3). Although the students had to work on only one project, changes of projects were expected during the early steps of the assignment. Usually the lack of timely feedback from the project team to the student was an indication that the assignment would be finished later. So, many students changed their projects in the early steps. However, the number of tested projects does not mean that students asked approval for each one of these projects on STEP 2, but rather that they used various projects before deciding which one to submit for approval.

Regarding their ability to be an equal part of the project community, both testers and developers said that it was easy for them to understand the work of others (Q7, Q10), while a significant number of testers said that, if needed, they could work as developers and fix the bugs reported by them (Q8) or by others (Q9).

Table 2: Students' responses.

	Testers (n = 89)	Developers (n = 27)	Engineers (n = 152)	Total (n = 268)
Q1.How many days passed before you started finding bugs/ writing code/ writing the SRS document? M (SD)				
	2.20 (0.76)	14.82 (16.72)	16.70 (15.02)	12.00 (14.26)
Q2. Was it easy for you to find an appropriate FLOSS project? Y | N (Y%)				
	24 | 65 (27%)	11 | 16 (41%)	59 | 93 (39%)	94 | 174 (35%)
Q3. How many projects did you try, before selecting the final one? M (SD)				
	11.09 (11.26)	5.55 (3.82)	4.74 (4.15)	6.76 (7.90)
Q4. Did you like working as a tester/ developer/ engineer of a FLOSS project? Y | N (Y%)				
	84 | 5 (94%)	26 | 1 (96%)	135 | 17 (89%)	245 | 23 (91%)
Q5. Was it easy for you to find a bug/ develop a functionality/ analyze a requirement? Y | N (Y%)				
	60 | 29 (67%)	14 | 13 (52%)	101 | 51 (66%)	175 | 93 (65%)
Q6. Was it easy for you to properly report a bug/ submit your code/ submit the document? Y | N (Y%)				
	79 | 10 (89%)	23 | 4 (85%)	103 | 49 (68%)	205 | 63 (76%)
Q7-Tester. Did you understand the bugs that other people reported on your FLOSS project? Y | N (Y%)				
	80 | 9 (90%)	n.a.	n.a.	80 | 9 (90%)
Q8-Tester. Can you fix the bugs that you found? Y | N (Y%)				
	39 | 50 (44%)	n.a.	n.a.	39 | 50 (44%)
Q9-Tester. Can you fix the bugs that other people found? Y | N (Y%)				
	29 | 60 (33%)	n.a.	n.a.	29 | 60 (33%)
Q10-Developer. Did you understand the code that other people submitted on your FLOSS project? Y | N (Y%)				
	n.a.	20 | 7 (74%)	n.a.	20 | 7 (74%)
Q11. Did you exchange messages with the project developers? Y | N (Y%)				
	50 | 39 (56%)	14 | 13 (52%)	106 | 46 (70%)	170 | 98 (63%)
Q12. Did you participate in discussions in project forums? Y | N (Y%)				
	28 | 61 (31%)	11 | 16 (41%)	52 | 100 (34%)	91 | 177 (34%)
Q13. Did you receive feedback through project forums/ emails/ private messages? Y | N (Y%)				
	58 | 31 (65%)	15 | 12 (56%)	73 | 79 (48%)	146 | 122 (54%)
Q15. Will you continue to participate in the FLOSS project after the completion of the assignment? Y | N (Y%)				
	63 | 26 (71%)	25 | 2 (93%)	94 | 58 (62%)	182 | 86 (68%)
Q16. Would you be interested in helping students next year by assuming the role of forum moderators, consultants, etc.? Y | N (Y%)				
	53 | 36 (60%)	14 | 13 (52%)	76 | 57 (57%)	143 | 106 (53%)

We have mentioned several times that the collaboration between students and the open source community was vital. This collaboration is depicted in the assignment through forum posts, messages, and other forms of discussions. More than half of the students exchanged messages directly with the developers of the project (Q11). The percentage was higher for the engineers, because they needed more

information to understand the project and produce a clearer SRS document. Participating in forum discussions was less appealing for the students (Q12), probably because they were more reserved and new members of the community. In general, a satisfactory number of students received feedback on their inquiries through various media (Q13). However, we would like to see this number rise in the future.

It is very encouraging to have a high percentage of students declare that they are going to remain members of the open source community, even after the completion of the assignment (Q15). This is a more real metric of the impact the approach had to our students. Even if the high percentages recorded drops after the assignment, the strong positive attitude is promising. Being members of an open, active, evolving programming community could be a goal in itself for the students of an Informatics department. The experience gained through this kind of work can be valuable for the professional development of the students.

Finally, students appreciated the value of past students' work. By reading what others did before them they get a useful depository of good and bad case scenarios. Additionally, students in the last two years were able to get feedback from past students who had the same assignment and the same issues to deal with. So, it is only natural that the majority of students expressed an interest to assume a consulting role, during the next academic year (Q16).

A number of open-ended questions were also included in the questionnaire, so that students could comment freely on improvement suggestions and issues they faced. Students asked for more support during the selection process. Although, support on how to find an appropriate project is already available, students felt that more guidance is necessary in order to identify a good project for the assignment. Another issue of concern for the students was the data organization inside the learning environment. Throughout the years the resources (e.g., documents, tutorials, etc.) and especially students' reports multiplied in volume. It was not always easy for the students to find what they were looking for. The most important issue regarding organization was that it was difficult to browse past reports. Although, the reports were organized by role, the student asked for more levels of organization, such as the type of the project (e.g., finance, game, etc.), the activity level of each project, etc. Additionally, they asked for faster response times from the instructor and the assistants in the learning environment. Finally, student praised the fact that they were able to work on real projects and experience the development of open source software first hand.

4 DISCUSSION

The hybrid method we apply may pose complex and difficult issues to the students. The main difficulty for the students is that for the first time during their studies, they are asked to work in the real context of a larger community. The results, however, showed that students are able to manage the activity and complete successfully the assignment, having a positive opinion for the method as a whole. Working in a real context is both a challenge and a motive for the students.

Regarding the issues raised by the students, a better organization scheme is needed, as is more support on the selection of a project. At this point, there is a review form and a rating scale available in all the submitted post, but students rarely used them. Although this does not cover students' demands on several levels of organization, the 5-star rating system could be used to distinct the good case scenarios from the rest. The fact that the review function is not used is an issue for us. There are many studies on how peer review can enhance learning in cognitive and meta-cognitive level (e.g., McConnell, 2001; Liu & Tsai, 2005; Papadopoulos et al., 2012). Even if the review comments are submitted to past reports, studies have shown that it is more beneficial for the students to submit many reviews than receiving comments (Lundstrom & Baker, 2009; Papadopoulos et al., 2012). Apart from strengthening analytical, comparing, and evaluating skills, a peer review process can be applied to support multiple perspectives. By conducting reviews the students get familiar with the work of their peers, get another view on the same issues and get the opportunity to see where the others are converging, thus identifying a dominant solution.

In the future, we intend to enhance the role of review in our method and better support students in peer review by guiding them using review guidelines and appropriate forms. This way we will give structure to the review (and drop the totally free mode that exists now) and make it more meaningful for the students.

Another issue that came up after three years of the course running is that there is a low level of collaboration between students of the same academic year inside the learning environment. Students tend to use the environment to get

feedback, but their questions are addressed to the instructor, teacher assistants, and past students acting as consultants. Log files and questionnaires show very limited interaction between peers. A better design is needed in order to strengthen the links between current students. For example, a student could be a member of a FLOSS community acting as a tester, and at the same time a member of the testers' community of the class. These smaller circles formed inside the students' cohort can provide additional assistance to the students and help them tackle collaboratively any common issue.

Finally, readers interested in applying similar method in their courses should take into account the instructor's overhead and the resource demands of the approach. Starting from the resources, the method utilizes pre-existing communities to what is called an outside approach of organization (Meiszner et al., 2009). However, a learning environment with basic communication tool-box is also needed. Although we used an all-inclusive environment, a combination of similar, freely available tools (forum, blog, wiki, etc.) could also be used. Regarding the instructor's overhead, the main task is the monitoring of students' progress throughout the different steps. This could be done by visiting students' personal blogs about their projects. Of course, the most intense period for the instructor is the second step of the process, when the students submit their projects for approval. This is not a job for one person, especially when we had 187 projects requiring approval in the third year. Right from the start of the first we formed a group of people that would assist students in their projects. Apart from the instructor, a number of teacher assistants and PhD candidates were enlisted to help. This group grew over the years, by including past students and external collaborators. However, students' demand for faster responds remained constant the last three years.

5 CONCLUSIONS

The use of FLOSS project in formal education has increasingly gained interest over the years. The benefits for the students could be multifold. However, attention is needed in designing a learning activity that utilizes the pre-existed communities of the open source world in a way that will not overwhelm the students.

In this paper we presented in detail an instructional approach that immerses students in software engineering through three different roles. Our intention is to keep using FLOSS projects as instructional tools in our software engineering courses. However, it is clear that improvements are needed in at least three areas. First, students need better support in selecting appropriate projects. Tutorial sessions and more detailed instructions could be useful for this. Second, we need to change students' attitude toward the peer review process. We need to clarify the purpose of such a process in the learning activity and help the students appreciate its benefits. Third, we need to enhance peer interaction, especially between students who work in the same context. In-class communities based on student roles can address this issue.

The results during the last three years were encouraging, showing that students are able to participate successfully in such an activity. Maybe the most promising finding was that students' expressed their intentions to remain members of their FLOSS projects communities, even after the completion of the assignment. The participation in such communities is important and could support students in skill development.

ACKNOWLEDGEMENTS

This work is partially funded by the European Commission in the context of (A) the OPEN-SME Open-Source Software Reuse Service for SMEs projects, under the grant agreement no. FP7-SME-2008-2/243768, (B) the openSE project under the grant agreement no. 503641-LLP-1-2009-1-PT-ERASMUS-ECUE, and (C) the FLOSSCom project under the grant agreement no. 229405 - CP -1-2006-1- PT - MINERVA – M.

REFERENCES

Fischer, G., (2007). Meta-design: Expanding Boundaries and Redistributing Control in Design. In *Proceedings of INTERACT 2007*. Rio de Janeiro, Brazil. 193-206.

Glott, R., Meiszner, A. and Sowe, S. K., (2007). FLOSSCom Phase 1 Report: Analysis of the Informal Learning Environment of FLOSS Communities", *FLOSSCom Project.* 2007.

Jaccheri, L., Osterlie, T., (2007). Open Source Software: A Source of Possibilities for Software Engineering Education and Empirical Software Engineering. *First International Workshop on Emerging Trends in FLOSS Research and Development*, 2007.

Liu, C. C. and Tsai, C. M., (2005). Peer assessment through web-based knowledge acquisition: tools to

support conceptual awareness. Innovations in Education and Teaching International, 42, 43-59.

Lundell, B., Persson, A., Lings, B., (2007). Learning Through Practical Involvement in the OSS Ecosystem: Experiences from a Masters Assignment. In *Proceedings of the Third International Conference on Open Source Systems 2007*, 289-294

Lundstrom, K. and Baker, W., (2009). To give is better than to receive: The benefits of peer review to the reviewer's own writing. *Journal of Second Language Writing*, 18, 30-43.

McConnell, J. (2001). Active and cooperative learning. Analysis of Algorithms: An Active Learning Approach. *Jones & Bartlett Pub*

Meiszner, A. Moustaka, K. and Stamelos, I., (2009). A hybrid approach to Computer Science Education – A case study: Software Engineering at Aristotle University. In: *CSEDU 2009 - International Conference on Computer Supported Education*, 23-26 March 2009, Lisbon, Portugal.

Meiszner, A., (2007a) "Communication tools in FLOSS communities: A look at FLOSS communities at large – Beyond the development team", paper and presentation at the *Web Based Communities Conference 2007*, Salamanca – Spain

Papadopoulos, P. M., Lagkas, T. D. and Demetriadis, S. N., (2012). How to Improve the Peer Review Method: Free-Selection vs Assigned-Pair Protocol Evaluated in a Computer Networking Course. Computers & Education (in press), doi: 10.1016/j.compedu. 2012.01.005.

Spinellis, D. (2006). Prof. Diomidis Spinellis, *Personal communication*, Athens, 2006.

Stamelos, I. (2008). Teaching Software Engineering with Free/Libre Open Source Projects. *International Journal of Open Source Software & Process (IJOSSP)*, Vol. 1(1), pp: 72-90.

Staring. K., Titlestad, O. H. (2006). Networks of Open Source Health Care Action. In the *Proceedings of the 2nd International Conference on Open Source Systems*, Springer-Verlag, 135-141.

Staring. K., Titlestad, O. H., Gailis, J. (2005). Educational transformation through open source approaches, IRIS'28 Meeting. http://wwwold.hia.no/iris28/Docs/ IRIS2028-1106.pdf