Electronic Communications of the EASST
Volume 33 (2010)

Proceedings of the
Fourth International Workshop on
Foundations and Techniques for
Open Source Software Certification
(OpenCert 2010)

Using Free/Libre Open Source Software Projects as E-learning Tools

Antonio Cerone and Sulayman K. Sowe

17 pages

Using Free/Libre Open Source Software Projects as E-learning Tools

Antonio Cerone[1*] **and Sulayman K. Sowe**[2]

[1] antonio@iist.unu.edu
UNU-IIST, Macau SAR China.

[2] sowe@ias.unu.edu,
UNU-IAS, Yokohama, Japan.

Abstract: Free/Libre Open Source Software (FLOSS) projects can be considered as learning environments in which heterogeneous communities get together to exchange knowledge through discussion and put it into practice through actual contributions to software development, revision and testing. This has encouraged tertiary educators to attempt the inclusion of participation in FLOSS projects as part of the requirements of Software Engineering courses, and pilot studies have been conducted to test the effectiveness of such an attempt. This paper discusses two pilot studies with reference to several studies concerning the role of learning in FLOSS projects and shows how using FLOSS projects as E-learning tools has a potential to increase the quality of the software product.

Keywords: OSS development; Education; Pilot Studies; Knowledge Exchange; E-learning; Software Quality

1 Introduction

Over the last years Free/Libre Open Source Software (FLOSS) communities have proven themselves to be able to deliver high-quality system and application software. Although FLOSS communities consist of heterogeneous groups of independent volunteers, who interact but are driven by different interests and motivations, and may appear to an external observer chaotic or even anarchic, they actually have specific organisational characteristic [Muf06]. These characteristics have been identified and analysed through empirical studies, which highlighted the implications of the FLOSS phenomenon throughout the information, knowledge, and culture economy, in a multidisciplinary context that goes well beyond software development [Muf06, Ben02]. Benkler [Ben02] goes even further and suggests reasons to think that peer-production may outperform market-based production in some information production activities in which a pervasively networked environment plays a major facilitating role. The generality of Benkler hypothesis makes it suitable to be applied to an educational context [Fut06]

Education has been showing during the last years multifaceted signs of crisis which affect all levels from primary to tertiary: diminishing academic achievements, increasing number of dropouts, teacher shortages and collapse of education reforms. A workshop held at Bagnols,

[*] *Correspondence Author*: Antonio Cerone. Email:{antonio@iist.unu.edu}.
Address: UNU-IIST, P.O. Box 3058, Macau SAR China. Tel: +853 2871-2930, Fax: +853 2871-2940

France, attended by educational practitioners, technologists, brain scientists and cognitive psychologists has identified factors in the current crisis in education and examined the potential uses of innovative technologies to support education [TS03]. Two important conclusions of the Bagnols workshop are that *education must be learner-centred* and that *learning must be social and fun* [TS03]. Learners are no longer comfortable with traditional modes of education, in which information is presented linearly, mostly in a text-based way, with almost no activities aiming to put acquired knowledge into real-life practise. This has created a mismatch between modes of education adopted by schools and universities and modern living style. In fact, nowadays information is presented to the public in daily life throughout multiple streams and multiple modalities simultaneously. The Internet provides a richer, much more frequently updated and more appealing source of information than printed newspapers, magazines and books. Moreover, information on the Internet is multi-modal and is organised in a tree-like or even graph-like structure rather than linearly. This allows learners to quickly *navigate* towards the targeted information in a way that appears to them more similar to pure entertainment than to academic work. Social relationships have been also heavily affected by the Internet: social networks, such as Facebook, allow individuals geographically distributed and with different cultural backgrounds to become friends, participate in online activities and games, join discussion fora and even establish romantic relationships.

FLOSS communities seem to have many characteristics that match the way information is best received by nowadays learners. They provide that sort of virtual world in which we often carry out our social and free-time activities. Moreover, FLOSS communities are natural instantiations of commons-based peer-production [Ben02, Ben07], the model of economic production in which the creative energy of large numbers of individuals is remotely coordinated, usually through the Internet, into large, meaningful projects mostly without traditional hierarchical organisation. Individuals participate in peer-production communities not just because of extrinsic motivations, such as solve problems, improve technical knowledge base, increase reputation and peer recognition and pass examinations, but also, and probably mainly, for a wide range of intrinsic reasons: they feel passionate about their particular area of expertise and enjoy self-satisfaction from sharing their knowledge and skills; they revel in creating something new or better; they have a personal sense of accomplishment and contribution and a sense of belonging to a community [Muf06, TW06, CS08].

FLOSS communities are therefore an ideal platform to implement learner-centred education in a social and fun manner, as envisaged by the Bagnols workshop, using the peer-production model, which has recently been taken as the basis on which to build new approaches to education [Fut06]. Although this approach can be potentially applied to any level [Fut06] and field of education [MGS09], this paper focuses on Software Engineering (SE) undergraduate and postgraduate courses [Kho09, SSD06, JØ07]. Application of FLOSS learning approaches to Software Engineering education is also a way of implementing the suggestion of the joint IEEE/ACM CS undergraduate curriculum guidelines [IEE04] that CS curricula should have significant real-world basis necessary to enable effective learning of software engineering skills and concepts.

All previous work in analysing learning aspects of FLOSS communities emphasises the benefits that the exploitation of such aspects may have on the educational process. In our work we also aim to identify the benefits that the explicit linkage of a FLOSS project to a formal education programme, such as a Software Engineering course or postgraduate research activity, brings to the FLOSS community itself and, in the end, to the quality of the FLOSS product.

In Section 2 we consider recent work that explores the link between FLOSS approaches and education [Fut06]. FLOSS communities are analysed as *collaborative networks* and *communities of practice* to extrapolate the learning process that facilitates the emergence and evolution of community members' knowledge [MGS09]. Challenges in adapting and transferring such a learning process to an educational setting are discussed.

Section 3 considers two research frameworks and corresponding pilot studies conducted to empirically analyse the use of FLOSS communities for formal education in Software Engineering at undergraduate [Sow08, Sta09, SGG, SSL06] and postgraduate [JØ07] levels. The two approaches are discussed with respect to the student's degrees of freedom (Section 3.3) and topical focus (Section 3.4). The proposal of a third pilot study [CS08] more ambitiously aims to operate changes into the structure and organisation of the FLOSS community to facilitate the use of innovative methodologies, such as formal methods, in which to involve students.

In Section 4 we show, with respect to Shaikh and Cerone's framework for evaluating quality of Open Source Software (OSS) [SC09], that the usage of FLOSS projects as e-Learning tools has the potential to increase the quality of the FLOSS product.

2 The Role of Learning in FLOSS Communities

One important attempt to identify a general link between FLOSS approaches and educational agendas is a 2006 report [Fut06] that looks at FLOSS as a cultural phenomenon and aims to extrapolate new approaches to teaching and learning and to define new models of innovation and software development in education. Drawing on Benkler's work on commons-based peer-production [Ben02] the report discusses strengths and weaknesses of FLOSS approaches which might apply to educational settings. Then it focuses on two ways in which peer-production FLOSS-like approaches may be used in teaching and learning:

collaborative network that is network that consists of a variety of entities that are largely autonomous, geographically distributed and heterogeneous in terms of their operating environment, culture, social capital and goals, but nevertheless collaborate to better achieve common or compatible goals and whose interactions are supported by computer network [CA06];

community of practice that is a group of people who share an interest, a craft, and/or a profession, which can evolve naturally because of the members' common interest in a particular domain or area or can be created specifically with the goal of gaining knowledge related to their field [LW91].

Distributed *collaborative networks* provide a powerful platform in which, due to the mediation of digital technology in a virtual environment, the duality teacher-learner fades out, and the two roles of teacher and learner merge together into the generic role of actor within the participatory culture of the network and its informal learning spaces [Fut06, MGS09]. From the learner's perspective, this enables the full range of potential intrinsic reasons mentioned in Section 1 to become actual motivations and to urge learners to play, alongside with teachers, their common role as actors in the community's activities. In addition, FLOSS communities are characterised by the freedom with which actors choose projects as well as the total control that actors have on the degrees of their own contribution to the project.

Freedom and equality of participants constitute a "democratic" basis for analysing FLOSS communities as *communities of practice*. Novices are always welcome by FLOSS communities, in which they undergo through a gradual process of social integration and skill development that allows them to earn a reputation as reliable developers and then move towards the leading positions in the community [Tuo05]. FLOSS communities are in this sense open participatory ecosystems [MGS08, MGS09], in which actors create not only source code but a large variety of resources that include the implicit and explicit definitions of learning processes and the establishment and maintenance of communication and support systems. Furthermore these resources are made visible and available to other actors. Therefore development (source code), support (tools) and learning (knowledge) emerge as the product of a continuous socialisation process in a virtual environment. Development of source code is enabled by building up knowledge about already produced code, through direct observation, review, modification as well as discussion with other actors, and about support tools, through direct interaction as well as access to documentation and discussion with other actors. As suggested by Sowe and Stamelos [SS08a] the learning process of individual actors can be divided in four phases through which knowledge evolves. We give our slightly different characterisation of such phases as follows:

socialise by *implicitly sharing knowledge*;

externalise *tacit knowledge* by making it explicit to the community;

combine community *explicit knowledge* and organise it as abstract knowledge;

internalise *abstract knowledge* by absorbing it and combining it with own knowledge and experiences to produce new tacit knowledge.

The four phases are not fully sequential but overlap in a certain measure, as shown in Figure 1. In particular, socialisation, after playing the role to initiate the learning process, is still active during the other phases for which it is actually the enabling factor.

If we want to transfer the learning process occurring within FLOSS communities to an educational setting, we need to better understand the cognitive aspects of the four phases above and interpret and implement them in a context driven by educational goals rather than just by software development.

Socialisation does not require an education-oriented interpretation and is probably the easiest phase to implement in an educational setting. In fact, socialising in a virtual environment, specifically through the Internet, already permeates our daily life and specific mechanisms and tools used by FLOSS communities, such as discussion fora, are general enough to be used for educational purposes; moreover, there are already several specific, and even more sophisticated (i.e. supporting multi-modal interation) e-learning tools and environments [Imm] that implement socialisation, such as Moodle [Moo] and Second Life [Sec].

Externalisation naturally occurs in an implicit way through socialisation tools such as discussion fora, but needs to be addressed by knowledge-management tools, such as repositories, to be effectively implemented in an explicit way. Tools used to manage and organise knowledge

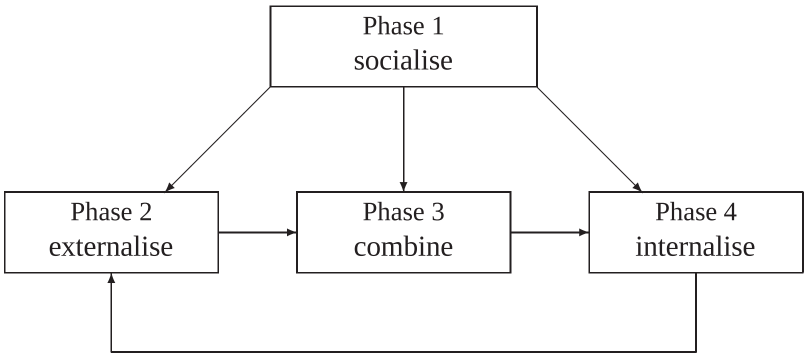

Figure 1: Learning process of individual actors in FLOSS communities

within FLOSS communities are often a challenge for the novice and actually require the user to go through a learning process before using them. Although this may be acceptable in a context purely driven by software development, in which skill in quickly acquiring familiarity with new tools may be considered a reasonable pre-requisite to enter the community, and may be even seen as a parameter to naturally select skilled contributors, the situation is totally different in an education-driven context. In such a context going through an heavy learning process to be able to use learning tools is definitely unacceptable. Therefore, existing tools have to be made more usable while more appropriate tools have to be developed to effectively implement externalisation in an educational setting. Externalisation is also intimately related to the intrinsic motivations of the user in joining the community and contributing to it. Intrinsic motivations, such as

- feel passionate about particular area of expertise,
- enjoy self-satisfaction from sharing knowledge and skills,
- have a sense of belonging to a community,

are all strong drivers for externalisation. There are also a number of extrinsic motivations that contribute to externalisation, which include

- solve particular technical problems/needs by exploiting Linus' Law: "given enough eyeballs, all the bugs are shallow" (from Linus Torvalds);
- public visibility to increase reputation and and peer recognition.

Combination of knowledge is incremental and consists of two main activities:

- multiple interactions with knowledge-management tools as well as with other members of the community to identify and extract relevant bits of explicit knowledge;
- combination and organisation of such bits of explicit knowledge to produce meaningful abstract knowledge.

The interaction with knowledge-management tools presents the same challenges as discussed for the externalisation phase. Organisation of explicit knowledge and production of meaningful abstract knowledge are cognitive activities within the ambit of *knowledge representation*. Several alternative theories have been proposed in cognitive psychology to explain knowledge representation within the human mind, but it is beyond the scope of this paper to deals with such theoretical aspects. From a pragmatic point of view we can say that the way individuals combine explicit knowledge is affected by the accessibility, structure and presentation of the contents of such knowledge and by own personal learning attitudes. Knowledge-management tools have therefore to address this issues as well as to enable individuals to have more control and responsibility for their learning [GFR$^+$05].

Internalisation of knowledge is a cognitive activity which is driven by both intrinsic and extrinsic motivations. *Intrinsic motivations* for internalisation are:

- revel in creating something new or better;
- have a personal sense of accomplishment and contribution.

Extrinsic motivations for internalisation are:

- improve technical knowledge base;
- pass examinations;
- develop the solution to a technical problem.

We will discuss in Section 3.4 how the grading approach utilised by the lecturer affects these extrinsic motivations and may create conflicts with intrinsic motivations, thus leading to a partial inhibition of internalisation. Internalisation is also facilitated by the efficacy and usability of code analysis tools such as bug trackers.

We will also discuss in Section 3.3 that limiting the degrees of freedom of students in participating in FLOSS projects may produce community members with little extrinsic motivations, with negative consequences for both the externalisation and internalisation phases of their learning process.

3 Frameworks and Pilot Studies in SE Education

The joint IEEE/ACM CS undergraduate curriculum guidelines [IEE04] suggest that CS curricula should have significant real-world basis necessary to enable effective learning of software engineering skills and concepts and should incorporate Capstone projects. Although many efforts have been made to involve students in software projects in local companies, most companies are not willing to sacrify their products to students who are constrained to complete the assigned work in one semester [Alz05]. In this scenario the *bazaar of learning* offered by FLOSS projects represents a meaningful alternative learning context to expose students to real-world software development activities [SSD06].

Characteristics and evolution modalities of FLOSS communities have been largely studied empirically by extracting data from repositories and performing statistical analysis on such data

[SII07]. However, learning aspects cannot be easily captured using this research methodology due to the absence of related information inside repositories. In this section, we focus on Software Engineering education and survey studies aimed to explore the use of FLOSS projects as e-learning tools.

During the last decade the FLOSS development model has deeply changed the way we develop and commercialise software, affecting traditional software development methodologies and posing serious challenges to commercial software industry. Students are strongly attracted by this new software development paradigm and enthusiastically join FLOSS projects. At the same time software industry is more and more including OSS skills and knowledge among their hiring selection criteria [Lon08]. This new scenario, in addition to the fact that FLOSS projects are actually Software Engineering practice, has made Software Engineering the most appropriate teaching subject to test the educational capabilities of FLOSS projects and has encouraged tertiary educators to attempt the inclusion of participation in FLOSS projects as part of the requirements of Software Engineering courses. Several pilot studies have been conducted to test the effectiveness of such an attempt and to assess the feasibility of full-scale studies.

3.1 Undergraduate Education Pilot Study

A pilot study conducted by Sowe and Stamelos [SS08b] addressed the open question as to whether the FLOSS methodology can be used to teach Software Engineering courses within a formally structured curriculum. The study was based on a pilot programme to teach software testing [SSD06] and aimed to develop and test a research method [SS08b] and to develop an approach to evaluate student participation [SSL06]. Within a pool of 150 undergraduate students enrolled in a course of “Introduction to Software Engineering” at Aristotle University, Greece, 15 joined the programme and 13 of them completed it. The study consisted of three phases in which students:

1. received lectures on FLOSS-related topics, browsed projects and selected one of them;
2. participated in the selected project with the aim to find and report bugs, and possibly fix them;
3. were evaluated and graded by the lecturers.

The study made use of two surveys in which students showed their interest in continuing their participation in the project after graduating. Student were actually forwarding responses from their projects to the lecturers after the pilot programme was ended and student grades published. This is a clear evidence that FLOSS projects can involve students in a long-term participation, which is in line with the need for life-long learning experiences, typical of a discipline in exponentially rapid evolution as is Software Engineering.

3.2 Postgraduate Education Pilot Study

Jaccheri and Østerlie [JØ07] use an approach for teaching master level students in which students are given assignments for which they have to

- survey literature on OSS development and formulate one or more research question(s) that could be addressed by participating in a project;
- select an OSS project which is appropriate for the assignment and the formulated research questions;
- act as developers in the selected project;
- act as researchers in the selected project by addressing the formulated research questions.

This approach has been used since 2002 by the Software Engineering Group (SU) [Con] of the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU). Jaccheri and Østerlie report on a concrete study based on this approach, in which one master student was requested to participate in a commercially controlled OSS project, the Netbean open source project, to understand how firms can benefit from using OSS [JØ07]. More specifically, the student was asked to determine how the use of Software Engineering techniques, such as explicit planning, ownership, inspection and testing, affects the OSS project. Within the scope of the assignment, the only constraint was to use action research (AR) [DMK04] as the methodology for the study. This study raises important considerations about the degrees of freedom given to the student. While students appreciate freedom in assignments as a positive learning experience, the authors recognise, as a result of the evaluation of their work by industrial professionals as well as discussions with other researchers, that it would have been more effective from a research perspective to provide students with predefined research questions. In the particular case study that they reported research questions were about the interaction between professionals and volunteers; this required the selection of a project in which commercial actors actively play significant roles. In an alternative research framework the project could have been selected before formulating the research questions. In an even more constraining framework the selection of the project could even been made by the lecturer.

3.3 Student's Degrees of Freedom

The degrees of freedom given to students is an important issue in both studies. In the first study [SS08b] undergraduate students joined the programme on a volunteer basis and had full freedom in selecting the project; given that the course was specifically about software testing, the assignment generically asked to find and report bugs, and possibly fix them. In the second study [JØ07] postgraduate students had full freedom in formulating research questions, but were constrained in selecting the project by their own choice of research questions.

One of the main reasons for the success of FLOSS projects is to be based on communities of volunteers who are totally free in choosing the way of contributing both in terms of tasks and time commitment. Intrinsic reasons are fundamental in motivating active and effective participation in a FLOSS project. Forcing the injection of actors who partly or entirely lack intrinsic motivations but are requested to play an active role in the community would not produce effective learning in those actors and may even be detrimental to the whole FLOSS project community. We have seen in Section 2 that the phases of learning that are heavily dependent on intrinsic motivations are externalisation and internalisation. These two phases include important cognitive activities and their incomplete actuation, as in case of lack of intrinsic motivations, severely inhibits the whole learning process.

It is therefore essential to preserve the volunteer-based approach while using participation in FLOSS projects for educational purposes, as it was done in the pilot study conducted at Aristotle University. In general, for undergraduate courses, we would suggest not to include participation in a FLOSS project as a course requirement unless the course is a very focussed elective. For postgraduate students, participation in a FLOSS project may be either related to a course or to a final thesis or project work. In general, it is expected that postgraduate students have more focussed interests and a higher degree of maturity than undergraduate students. In this perspective, a postgraduate student who has chosen an elective course or a thesis topic which requires participation in a FLOSS project is supposed to have sufficient intrinsic motivations to succeed in the task.

The issue of the project selection is a very subtle one. In both pilot studies described above the project selection is left to the student, although some general selection criteria are provided. However, in the pilot study conducted at the Norwegian University of Science and Technology the selection of the project strongly depends on the research questions previously formulated by the student. The fact that the student has chosen a specific research question does not exclude that such research question may rule out all projects in which the student is likely to be enthusiastically interested. In this sense a research framework in which the project is selected before formulating the research questions is more sensible. In general, in designing the study research framework it is essential to ensure that there are no requirements for the student that may explicitly or implicitly reduce the student's degrees of freedom in choosing the project.

3.4 Student's Topical Focus

In the two pilot studies described in Sections 3.1 and 3.2 the student's topical focus in participating in the project was dictated by the assignment. In the pilot study conducted at Aristotle University the student had to find and report bugs, and possibly fix them. The grading system included marks for email exchange with the lecturer concerning the project, proper use of bug tracking system or bug database and testing activity measured by the number of bugs found, reported and fixed, and by the number of replies to the reports. This restricted focus has probably worked as an extrinsic motivation that prevented students from contributing to the project in terms of software development, for which there was no mark. As a result students probably felt that the effort needed to fix bug was not sufficiently rewarded in terms of marks. This hypothesis is confirmed by one outcome of the study: although students performed well in finding and reporting bugs, they did not well in fixing bugs [SSL06].

It is inevitable that the grading and evaluation approach strongly affects student's extrinsic motivations: the more transparent and explicit the grading approach the stronger the effect on extrinsic motivations. A grading approach that has a strong effect on extrinsic motivations does not allow students to achieve a complete involvement in the project and often causes a conflict with intrinsic motivations, which are an essential driver in FLOSS project. Such a conflict may result in an incomplete actuation of the internalisation phase of the learning process, which depends on both intrinsic and extrinsic motivations, and may inhibit potential learning capabilities of the student.

However, avoiding such a strong effect is not easy in a formal education context. Quantifying the evaluation of the participation in the project as a whole with no further details would not be a feasible solution. Such a solution would be clearly against usual university policies that require lecturers to make the grading approach public by quantifying each contribution in terms of percentage of the final grade. Moreover, limiting the information about the assessment procedure that is provided to students would be inherently unfair and might promote suspicion among students. And in the end, this would actually reduce extrinsic motivations of students. Possible solutions to the problem could be that lecturers

- evaluate the participation in the project indirectly by assessing a written report and publish details of the grading of such report;
- discuss beforehand the grading approach with the students and agree on the details with them;
- provide alternative assessment and/or grading approaches among which the students may choose;
- develop an appropriate peer assessment approach.

These proposed solutions are neither exhaustive nor mutually exclusive.

In the pilot study conducted at the Norwegian University of Science and Technology, which involved postgraduate students, the student's task was not only to actively participate in the project, but also to use such participation to address research questions previously formulated. This is an interesting attempt to involve learners in studying and possibly improving the FLOSS development process, that is, studying and possibly improving the learning tool (i.e. the FLOSS project) they are using. In the Norwegian study the student's focus was on management and organisational aspects of Software Engineering.

There are other aspects of Software Engineering in which students, especially postgraduate students, may contribute, through their participation in a FLOSS project, to provide new insight in relation to the FLOSS approach. One of these aspects is *quality assurance*. The lack of central management in FLOSS projects makes it difficult to define a standard that could suggest indicators of the technical rigour used by a distributed community of volunteers and identify the human processes involved in the project [Mic05, MHP05]. Without precise indicators of this sort we cannot produce an effective quality assurance methodology for the released software. Zhao and Elbaum [ZE00] conducted a survey to examine the factors underlying quality assurance methods used within FLOSS communities and found out that their general attitude and practices towards quality and realising quality assurance practices are somewhat different to those prevalent in traditional software development. This situation opens a lot of research questions which could be addressed in studies conducted by postgraduate students through their involvement in FLOSS projects. In Section 4 we will further discuss the impact that such involvement could have on the quality of FLOSS products.

Postgraduate students are often exposed during their study to innovative software design and analysis technologies that enjoy little appreciation outside the academic world, either because such technologies are not mature enough to be applied to practical projects or because, in an industrial perspective, their cost prevail on the actual benefit they bring. Formal methods are one of such innovative technologies. Postgraduate students could bring new insights in FLOSS communities through the application of new specification and verification technologies such as formal modelling, model-checking and theorem-proving. This would require students to reverse engineer FLOSS code into a formal model and apply formal techniques to analyse the model. Unfortunately most FLOSS developers are unlikely to be familiar with formal methods and probably view them with a similar reluctance as does the industrial world. Feeding results of formal analysis back to the FLOSS project would be therefore a big challenge for the students. Here a soft approach would be needed: outcomes of formal analysis should be mapped back to code and test cases before been presented to the community. To this purpose, formal modelling techniques that provide counterexamples when a required system property is proven not to hold, such as model-checking, are the most appropriate. Besides the soft approach, it would be important to include in bug reports some information about the formal results that led to the bug identification. In this way, students would play the role of educators in their interaction with FLOSS developers, so fostering a gradual acceptance of new technologies by the FLOSS communities.

An alternative approach to promote the use of formal methods in the FLOSS community is a pilot project proposed by Cerone and Shaikh [CS08] as an attempt to explicitly introduce formal methods in the FLOSS development process. The most difficult task in this attempt is to preserve the intrinsic freedom that characterises contributions by the volunteers who join FLOSS projects. In fact, it would not be acceptable, and neither would it be accepted by the FLOSS community, to explicitly enforce the use of a specific formal modelling framework to be adopted by all project participants. In order to support open participation and, consequently, bottom-up organisation and parallel development, the project should therefore introduce and present formal methods only as a possible but not mandatory option available to the contributors. This approach would require an additional effort by the project leader team in facilitating the integration of those contributions that do not make any use of formal methods into the new development model. An important role would be played here, once again, by postgraduate students called to reverse engineer code, produced by other actors in a traditional FLOSS way, into changes and extensions to the formal model.

4 Impact on the Quality of FLOSS Products

We have seen in Section 3 that students can successfully use FLOSS projects as e-Learning tools and gain effective learning of software engineering skills and concepts from participating in FLOSS project. We have also seen that students, and in particular postgraduate students, can produce important contributions to the evolution of the FLOSS development model. In this section we investigate how such contribution can actually have impact on the quality of FLOSS products.

Shaikh and Cerone [SC09] have identified some factors that are unique to the FLOSS development process and influence the entire software development process and, consequently, the quality of the final software product. In their work, Shaikh and Cerone also define an initial framework in which such factors can be related to each other and to the quality. In particular, they distinguish three main notions of quality in the context of FLOSS development

quality by access which aims to measure the degrees of availability, accessibility and readability of source code in relation to the media and tools used to directly access source code and all supporting materials such as the documentation, review reports, testing outcomes, as well as the format and structural organisation of both source code and supporting materials.

quality by development which aims to measure the efficiency of all development and communication processes involved in the production, evolution and release of source code, its execution, testing and review, as well as bug reporting and fixing;

quality by design which corresponds to the traditional notion of software quality [IEE99, Pre00]: the end quality is judged by the design and implementation of the actual software and the code that underlies it.

Quality by access would greatly benefit from the use of formal methodologies by postgraduate students participating in the project. The reverse engineering of FLOSS code into formal models improves understanding the system architecture and the structure of code and leads to the production of better documentation. A by-product of the reverse engineering process is also the identification of inconsistencies and redundancies in the code and, as a consequence, its improvement with an increase in readability. Formal verification techniques produce results that are more general and understandable than the ones obtained using traditional testing techniques. Moreover, these results can be tracked back to the model, facilitating the fixing of bugs.

We have seen in Section 2 that availability and usability of knowledge-management tools is essential to enable the externalisation phase of the learning process. The development of new tools and the improvement of usability in existing tools with the aim to address the learning process in FLOSS communities is therefore likely to increase quality by access.

Quality by development is an attempt to measure the efficiency of all processes aiming to produce and review code and the interaction between them. Shaikh and Cerone [SC09] identifies five factors on which this notion of quality depends:

- precise and explicit understanding of software goals and requirements;
- choice of methodologies for testing, debugging and error and bug reporting;
- choice of programming languages and development environments;
- tools to provide effective communication, coordination and overall management of the project;
- facilitation of rapid frequency of beta releases.

We observe that the usage of FLOSS projects as e-Learning tools has the potential to affect these factors in a way that increases quality by development. First we observe that an additional effect of reverse engineering FLOSS code into formal models is the explicit definition of software requirements. Second, we believe that if methodologies, programming languages and tools are chosen having in mind not only their usage in software development but also their educational values, then there is a positive impact on the entire project community and, as a result, an additional benefit for the development process. Third, the frequent injection of students with short deadlines to complete their assignments may facilitate rapid frequency of beta releases. Finally, empirical studies such the one presented in Section 3.2 can produce a better insight in how these factors interact with each other and affect each other in the global context of the project management and organisation.

Quality by design, the traditional notation of quality, can be seen in the FLOSS context as a specific measure of

- the use of recognised software design notations, formal notations and analysis techniques to provide correctness with respect to explicitly desired safety, security and non-functional properties, and
- the production and frequent update of appropriate and explicit documentation that helps both the users and future developers.

We have seen in Section 3.3 that student participation can bring innovative software design and analysis technologies, such as formal methods, into FLOSS projects, thus increasing the community knowledge and, on the long term, increasing the acceptance of these technologies within FLOSS communities. Moreover, pilot projects aiming to explicitly incorporate these technologies in the FLOSS development process [CS08] could show whether or not there is an effective increase in quality by design. As for documentation, it is likely that student participation would increase its production, since written reports to document code production and performed analysis are a common form of assignment.

Finally, as we have anticipated in Section 3.3, postgraduate students may contribute, through research-driven participation in a FLOSS project, to identify quality indicators and define quality metrics appropriate for the FLOSS development model.

5 Conclusion and Future Work

In this paper we have considered recent work that explores the link between FLOSS approaches and education and described the dynamics of the learning process that facilitates the emergence and evolution of community members' knowledge. We have then considered two pilot studies conducted to empirically analyse the use of FLOSS communities for formal education in Software Engineering, discussed choices made in designing the research frameworks for the two studies and proposed suggestions to improve the frameworks to better match the student's learning process. Finally, we have shown that the use of FLOSS projects as e-Learning tools has a potential to increase the quality of the software product.

This research has been conducted as a preliminary analysis towards the objective of building a worldwide university network, coordinated by the United Nations University (UNU), to implement the use of FLOSS projects as e-Learning tools in Software Engineering postgraduate education. A first step in our future work is to design a framework, which incorporates the recommendations we presented in Section 3, for geographically distributed pilot studies in which students

- are totally free in the choice of the FLOSS project;

- are evaluated using a grading approach that is not likely to weaken their intrinsic motivations and that possibly strengthen their extrinsic motivations;
- are requested to participate in the project they have chosen but are totally free in choosing the form of participation;
- may have various levels of time commitment, which correspond to distinct numbers of credits;
- may choose, on a volunteer basis, a focus for their participation in the project among different topical areas such as code development, review, testing, reverse engineering, formal analysis;
- may choose, on a volunteer basis, a research topic concerning the investigation of the FLOSS phenomenon, which may include learning, project management, communication, social aspects, software quality, etc.

A second step is the creation of *pilot projects* in line with Cerone and Shaikh proposal [CS08], with academics and former students who have taken part in the pilot studies of the first step, being part of the leader team.

The final objective is to build a *postgraduate e-Learning programme* in OSS approaches to Software Engineering as part of the new UNU postgraduate programmes, and utilise some of the most successful pilot projects as e-learning tools within such a programme.

Bibliography

[Alz05] Z. Alzamil. Towards an Effective Software Engineering Course Project. In *Proceedings of the 27th International Conference on Software Engineering*. Pp. 631–632. ACM Press, 2005.

[Ben02] Y. Benkler. Coase's Penguin, or, Linux and The Nature of the Firm. *The Yale Law Journal* 212:369–446, 2002.
URL: http://www.yalelawjournal.org/images/pdfs/354.pdf.

[Ben07] Y. Benkler. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press, 2007.

[BTD05] J. M. Barahona, C. Tebb, V. Dimitrova. Transferring Libre Software Develoment Practises to the Production of Educational Resources: the Edukalibre Project. In *Proceedings of the 1st International Conference on Open Source Systems (OSS2005)*. Genova, Italy, 11–15 July 2005.

[CA06] L. M. Camarinha-Matos, H. Afsarmanesh. Collaborative networks: a new scientific discipline. *Journal of Intelligent Manufacturing439452*, 16:439–452, 2006.

[Con] R. Conradi et al. Software Engineering Group homepage. Norwegian University of Science and Technology (NTNU). URL: http://www.idi.ntnu.no/grupper/su/.

[CS08] A. Cerone, S. A. Shaikh. Incorporating Formal Methods in the Open Source Software Development Process. In *Proceedings of the OpenCert and FLOSS-FM 2008 joint Workshop*. UNU-IIST Research Report 398. 2008.

[DMK04] R. M. Davison, M. G. Martinsons, N. Kock. Principles of Canonical Action Research. *Information Systems Journal (ISJ)* 14(2):65–86, 2004.

[Fut06] Futurelab. The potential of open source approaches for education. Opening Education Report. Published online, 2006.
URL: http://www.futurelab.org.uk/resources/publications-reports-articles/.

[GFR+05] H. Green, K. Facer, T. Rudd, P. Dillon, P. Humphreys. Personalisation and Digital Technologies. Opening Education Report. Published online, 2005.
URL: http://www.futurelab.org.uk/resources/publications-reports-articles/.

[IEE99] IEEE Std 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology. February 1999.

[IEE04] IEEE/ACM Joint Task Force on Computing Curricula. Software Engineering 2004 Curriculum guidelines for Undergraduate Degree Programs in software Engineering. 2004. URL: http://sites.computer.org/ccse/SE2004Volume.pdf.

[Imm] Immersive Education. URL: http://immersiveeducation.org/.

[JØ07] L. Jaccheri, T. Østerlie. Open Source Software: A Source of Possibilities for Software Engineering Education and Empirical Software Engineering”. In *Proceedings of the Workshop on Emerging Trends in FLOSS Research and Development, co-located at ICSE’07*. Minneapolis, US, 21 May 2007.

[Kho09] A. Khoroshilov. Open Source Certification and Educational Process. In *Proceedings of the 3rd International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2009)*. Electronic Communications of the EASST 20. 2009.

[Lon08] J. Long. Open Source Software Development Experiences on the Students’ Resumes: Do They Count? - Insights from the Employers’ Perspectives. *The Journal of Information Technology Education (JITE)* 8:229–242, 2008.

[LW91] J. Lave, E. Wenger. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, 1991.

[MGS08] A. Meiszner, R. Glott, S. K. Sowe. Free/Libre Open Source Software (FLOSS) Communities as an Example of successful Open Participatory Learning Ecosystems. *The European Journal for the Informatics Professional, UPGRADE* IX(3):62–68, 2008.

[MGS09] A. Meiszner, R. Glott, S. K. Sowe. Preparing the Ne(x)t Generation: Lessons learnt from Free/Libre Open Source Software — Why free and open are pre-conditions and not options for higher education! In *Proceedings of the 4th International Barcelona*

Conference on Higher Education. Volume 2. Knowledge technologies for social transformation. Barcelona, Spain, 15–19 July 2009.

[MHP05] M. Michlmayr, F. Hunt, D. Probert. Quality Practices and Problems in Free Software Projects. In Scotto and Succi (eds.), *Proceedings of the First International Conference on Open Source Systems*. Pp. 24–28. Genova, Italy, 2005.

[Mic05] M. Michlmayr. Quality Improvement in Volunteer Free Software Projects: Exploring the Impact of Release Management. In Scotto and Succi (eds.), *Proceedings of the First International Conference on Open Source Systems*. Pp. 309–310. Genova, Italy, 2005.

[Moo] Moodle. URL: http://moodle.org/.

[Muf06] M. Muffatto. *Open Source — A Multidisciplinary Approach*. Imperial College Press, 2006.

[Pre00] S. R. Pressman. *Software Engineering - A Practitioner's Approach*. McGraw-Hill International, London, 2000.

[SC09] S. A. Shaikh, A. Cerone. Towards a Metric for Open Source Software Quality. In *Proceedings of the 3rd International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2009)*. Electronic Communications of the EASST 20. 2009.

[Sec] Seconf Life. URL: http//secondlife.com/.

[SGG] S. K. Sowe, R. A. Ghosh, R. Glott. A Model for Teaching and Learning in Open Source Software Projects: Constructivist Approach. Submitted for publication.

[SII07] S. K. Sowe, G. S. Ioannis, M. S. Ioannis (eds.). *Emerging Free and Open Source Software Practices*. IGI Global, 2007.

[Sow08] S. K. Sowe. Pilot Studies Relevant to Learning in FLOSS. In *Proceedings of the 1st International Conference on Free Knowledge, Free Technology (FKFT)*. Education for Free Society, Barcelona, Spain, 15–18 July 2008. URL: http://www.slideshare.net/andreasmeiszner/pilot-studies-relevant-to-learning-in-floss.

[SS08a] S. K. Sowe, I. Stamelos. Reflection on Knowledge Sharing in F/OSS Projects. In *Open Source Development, Communities and Quality*. IFIP International Federation for Information Processing 275, p. 351358. 2008.

[SS08b] S. K. Sowe, I. G. Stamelos. Involving Software Engineering Students in Open Source Software Projects: Experiences from a Pilot Study. *Journal of Information Systems Education (JISE)* 18(4):425–435, 2008.

[SSD06] S. K. Sowe, I. Stamelos, I. Deligiannis. A Framework for Teaching Software Testing using F/OSS Methodology. In *Proceedings of the 2nd International Conference on Open Source Systems (OSS2006)*. Como, Italy, 8–10 June 2006.

[SSL06] S. K. Sowe, I. G. Stamelos, A. Lefteris. An Empirical Approach to Evaluate Students Participation in Open Source Software Projects. In *Proceedings of the the IADIS CELDA conference*. Barcelona, Spain, 8–10 December 2006.

[Sta09] I. Stamelos. Involving Software Engineering Students in Open Source Software Projects: Experiences from a Pilot Study. *International Journal of Open Source Software & Processes (IJOSSP)* 1(1):72–90, 2009.

[TS03] M. Tokoro, L. Steels. *The Future of Learning*. IOS Press, 2003.

[Tuo05] I. Tuomi. The future of open source: Trends and prospects. In Wynants and Cornelis (eds.), *How open is the future? Economic, social and cultural scenarios inspired by free and open source software*. Pp. 429–459. Vrjie Universiteit Press, 2005.

[TW06] D. Tapscott, A. D. Williams. *Wikinomics: How Mass Collaboration Changes Everything*. Portfolio Books, 2006.

[ZE00] L. Zhao, S. Elbaum. A survey on quality related activities in open source. *ACM SIGSOFT Software Engineering Notes* 25(3):53–57, May 2000. ACM Press New York, NY, USA.