Leveraging Free and Open Source Software to Model Teaching and Learning in Computer Science Education

Sulayman K. Sowe, *Member, IACSIT*

***Abstract*—Free and Open Source Software (FOSS) is not only seen by computer science departments as an optional free and low cost technology to support their IT infrastructure and administrative duties, but also as a methodology to improve pedagogy and provide opportunities for students to learn. As a result, many teachers are turning to FOSS projects and communities as effective tools to teach computer science courses and give their students significant real-world experience. However, few teaching and learning models exist that can help computer science teachers provide guidance and mentorship for their students' involvement in FOSS projects. In order to address this gap in computer science education, this paper presents a teaching and learning model that was evaluated using a series of pilot studies conducted with software engineering students. Experiences and lessons learnt from the pilot studies shows one possible way teachers can guide students' involvement in FOSS projects and how this approach can be integrated into a formally structured curriculum.**

***Index Terms*—Capstone projects, computer science education, curriculum design, open source software communities, open source software projects, software engineering.**

I. Introduction

Curricular often emphasize that computer science in general and software engineering (SE) courses, in particular, should have a significant real-world experience necessary to enable effective learning of software engineering skills, processes, techniques, and concepts. In principle, didactics should challenge, motivate and engage students in meaningful and realistic activities so that they go out to become useful members of society. Many curricular have been designed with these novel principles in mind. The IEEE/ACM Joint Task Force on Computing [1] shows what to teach, how to teach what, and the expected outcomes of computer science and SE courses.

Computer science (CS) teachers are making efforts to give students practical hands-on experience by, for instance: (i) assigning students to software projects in local companies; (ii) exposing students to SE practice and tools through capstone projects [2]; (iii) guiding and mentoring students in their own engineering projects; (iv) providing environments for students to collaborated with their peers across geographical boundaries [3], exchange visits [4]; (v) enabling face-to-face collaboration between small groups of engineering students.

Yet, many CS or engineering graduates are ill-prepared to face and work in realistic software engineering environments. One reason for this, argued [5], might be due to the fact that software projects in the real-world involves participants with different skills and experiences, are full of inconsistencies, complex in terms of both social and technological dynamics, takes time to develop (usually spanning longer than a school semester), and are changing all the time. These characteristics are not only challenging but difficult to replicate in many classrooms with fixed time-tables and short semesters.

A. Research Contribution

There is increasing educational interest in Open Source projects as "bazaars of learning" [6]. In this learning milieu, students can be involved in exploratory and meaningful or realistic activities [7]. When FOSS is used as an alternative teaching methodology, students can be involved in meaningful software development activities, have experience in dealing with realistic software systems [8], access large quantities of code written by professional software developers, work with up-to-date software engineering tools, and collaborate with a diverse group of individuals from different parts of the world [9].

However, teaching in FOSS like environments is disadvantaged because there are no guidelines or models for educators to follow. CS teachers are still experimenting with different teaching methods, trying-out what works and what does not work, and consulting with peers. What seem to be missing in FOSS education is a teaching and learning model that can help teachers guide and mentor students involvement in FOSS projects. Thus, the aim of this paper is to fill this gap in the CS literature and contribute to FOSS education by presenting a teaching and learning model grounded on the principles of constructivist learning.

II. Background and Related Work

Computer science graduates should be able to cope with and even benefit from the rapid changes in the computing field. FOSS development is a rapid changing phenomenon which is fundamentally changing the way traditional software is being developed, distributed, maintained, and supported. Such a change might not be the silver bullet to today's educational problems, but educators are looking towards the FOSS methodology as a meaningful teaching and learning alternative [10]. The impetus is that students can learn software engineering concepts and skills, and be "better" prepared for today's world of work and actively

Manuscript received December 8, 2011; revised December 27, 2011. This work was supported by the Japan Society for the Promotion of Science (JSPS) under Grant P10806.

Dr. Sulayman K. Sowe is with the United Nations University Institute of Advanced Studies, Yokohama 220-8052, Japan (e-mail: sowe@ ias.unu.edu).

participate in globally distributed software development projects. Various forges (e.g Schoolforge.net), projects and non-profit foundations (the GNOME educamp project) are providing case studies, scenarios, guidelines, and open educational resources for students, teachers, and decision-makers. These initiatives many help educators experiment and embed FOSS like learning within their formal educational environments. In FOSS projects, both students and teachers have 'free' access to tools necessary for learning without having to pay for licenses fees to acquire and use those tools whilst in school, at their homes, and in their workplace when they graduate.

A. Constructivist Didactics in FOSS

The principles of constructivist learning [11] can be identified in many FOSS projects and are the guiding principles in the design of the model for teaching and learning in projects. Constructivist theory of learning plays great emphasis on the active and autonomous role for the learner [12]. Meaningful learning in FOSS projects takes place through scaffolding [13] - a form of tutorial assistance in which the learner interacts with expert community of software developers and users. One's ability to learn in this environment is intimately associated with his/her connection with others, in the real as well as the virtual world. FOSS development accommodates this social aspect of learning and uses technological artifacts (CVS/SVN, Mailing lists, and Bug trackers), interaction with others both within and across projects, and the application of domain knowledge as an integral aspect of learning.

Learning in FOSS means constantly reflecting on what has been discussed in mailing lists and forums, adjusting experiences, and accommodating that of others. However, we must acknowledge that in order to effectively learn from FOSS projects and communities, the learner must be actively involved in the learning process, be motivated to contribute his prior knowledge and experience, and be prepared to accommodate the view of others. The learner needs to revisit ideas, ponder, and experiment. She/he must be flexible, keep abreast with project's activities by constantly checking and tracking code commits, monitoring and frequenting forums and bug tracking systems to see which bugs are open, closed, fixed, or feature requests filed by newcomers.

III. A Model for Teaching and Learning in FOSS Projects

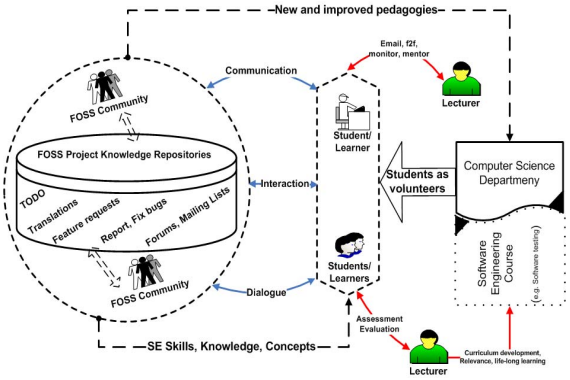

Fig. 1. The open source teaching and learning model.

The Open Source teaching and learning model in fig. 1 helps educators extend the methodology by which they teach and guide students towards meaningful and lifelong learning of software engineering concepts. The aims of the model are threefold:

- provide opportunities for students to work on what they consider interesting;
- give students real world experience in dealing with large software projects; and
- Opportune students to work, collaborate, and share their knowledge and experience with colleagues in distributed software development projects.

The model depicts an FOSS like learning environment which allows CS departments or lecturers to send their students (volunteers) out to participate in various software development activities (e.g. software testing, documentation, localization, coding, etc.) in FOSS projects. The students communicate, interact, and enter into dialogue with fellow software developers and users, and acquire essential software engineering skills.

In many mature projects, a knowledge repository also provides the tools for students to participate and contribute to the project's activities. Students can be involved in a host of activities in TODO lists, translate the software in their own language, test Alpha and Beta versions of the software, make feature requests, etc. Furthermore, computer science department also benefits from students' involvement and can use feedback received from students and projects to help them design new or improve their existing teaching and learning. The role of the lecturer is that of constant mentoring and monitoring of students through email conversations or face-to-face (f2f) interactions, assessing and evaluating students' performance in the projects. The lecturer can then use his experience to update his teaching syllabus, making it more relevant to support students in their lifelong learning experience.

A. Model Evaluation

The Open Source teaching and learning model was used as a didactic tool to implement three pilot studies with computer science undergraduates specializing in an introductory course to software engineering course. Table 1 shows the number of students involved in each activity type during each pilot study. Each pilot was conducted in three phases.

TABLE 1: Pilot Studies as Inputs to FOSS Teaching and Learning Model.

Pilot #	Learning Activity Type	Number of Students
1	Software Testing	13
2	Software Testing	22
	Software Development (Coding)	2
3	Software Testing	12
	Software Requirement Analysis	21
	Software Development (Coding)	3

1) Phase 1:

In phase one, students were introduced to Open Source software development, as well as the composition and dynamics of FOSS projects and communities. The students also learnt how to use *de facto* FOSS development and

collaboration tools such as CVS/SVN, bug tracking systems, and mailing lists. Online communication etiquettes were also discussed to help students "ask questions the smart way" [14]. During this three weeks introductory course, students also explored and discussed how popular Open Source software projects (eg. Mozilla) work and source code repositories and portals or forges such as SourceForge.net, Tigris.org, and Freshmeat.net. Projects selection criteria were developed to guide the students in selecting a project to work on. In choosing a project, students were asked to pay particular attention to the following criteria:

- Intended Operating system (Linux, Windows, or MAC) and programming language used to develop the software.
- The number of developers registered in the project and how active the forums and mailing lists are
- The development status of the project. Is it at the Alpha, Beta, or Mature stage?

2) Phase 2:

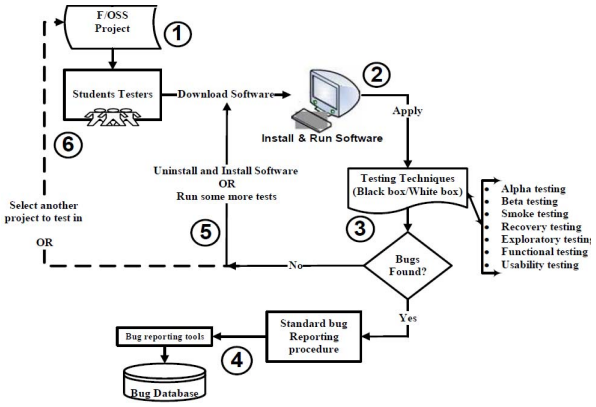

Fig. 2. Software testing activity

In phase two, which lasted for about eight to nine weeks, the students selected their projects and began their learning activities in the areas of software testing, requirement analysis, and software development or coding. For example; Fig. 2 shows how the software testing activity proceeded. A student selects a project of his/her choice (1), download and install the software in his/her computers (2). The student is also supposed to frequent the project's website and forums to monitor what is being discussed about the project. The student then applies software testing techniques (3), already learnt in previous lessons, on the installed software. If a bug is found, the student reports the bug to the project's community by filing a bug report (4). However, if no bug is found, the student can carry out more tests on the same software or install a newer version (5). There is no limited to the number of projects students can work on. Therefore, at any given time they can select another project to work on (6).

In software requirement analysis (SRA), students were given a predefined SRA template which they can modify depending on their project's requirements. Students first inspect the project website, documents, wikis, etc. to see if there are any functional and non-functional requirements specifications for the software. If none exist, they then suggest to the project community that they would like to contribute SRA for the project.

In software development, students can either take part in direct coding activities (checking-out, modifying, and committing bits of code to the projects code repository), or develop their own project and host it as Open Source in one of the FOSS portals, e.g. Sourceforge.net.

3) Phase 3:

In phase three, which lasted for about 1 week, the students presented and discussed their work in class. They were then graded for their performance and asked to complete a survey, which aimed to capture their motivation and experience in working in FOSS projects.

IV. Results and Discussions

A. Software Testing

In software testing, 47 students tested alpha and beta versions of the software in more than 70 projects. As shown in table 2, the students reported 530 bugs (mean = 11.28; Std. deviation = 5.702), which generated 455 replies or comments from their respective projects' communities (mean = 9.68; Std. deviation = 6.963). Bug reporting was highly correlated (r = 0.551; p=0.000) with replies received from their respective projects.

TABLE 2: Descriptive Statistics of Testing Activity

		Bugs reported	Replies received
N	Valid	47	47
	Missing	0	0
Mean		11.28	9.68
Median		10.00	8.00
Std. Deviation		5.702	6.963
Min.		3	1
Max.		25	31
Sum		530	455

B. Software Requirement Analysis (SRA)

During pilot 3, each of the twenty-one students wrote a SRA document for one of his/her project. Two documents were incomplete and 19 were completed. For the latter, 13 were accepted for publication in the projects' web sites.

C. Software Development (Coding)

During pilots 2 and 3, five students were involved in programming. One of them took part in coding and posted tips on configuration details of pulseaudion and other sound output modules in the Ubuntu project. One of the students became an active contributor with commit privileges in the Jajuk project. The other students collaborated to initiate and host the webtooy project at sourceforge.net, as shown in the screenshot in fig. 3.

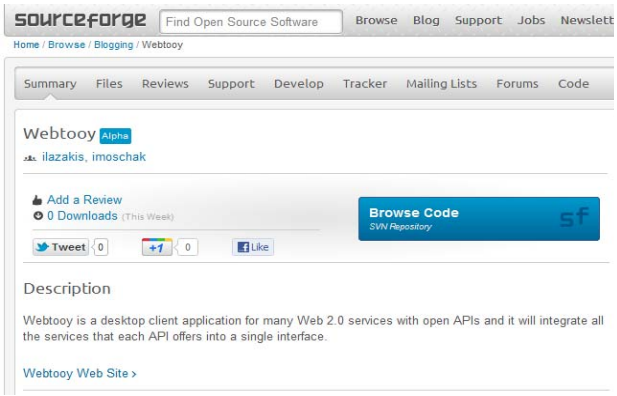

Fig. 3. Students project (webfoot) at sourceforge.net

V. Experience and Lessons Learnt

A. Students' Experience:

Students expressed satisfaction that they had a very interesting experience working in FOSS projects; engaging software developers and users; having the opportunity to express themselves in forums and mailing lists; cherished seeing their contributions appreciated almost instantly; and receiving encouragement from people they don't know and wouldn't have met face-to-face. However, learning in FOSS comes with a high cost. Some of the students complained that they do not have the skills to work in some projects. Some could not find interesting project to work on –there are too many FOSS projects, and some developers were not responding to their queries. Some comments students made about their involvement in FOSS projects buttresses these points.

"Some programs also required me to install other programs before I could run them. Many projects were 'dead' or inactive... Not so many active and beta testing projects available that am interested in... Most of the projects I tried did not address my needs".

Other barriers for students' effective participation in open source projects have to do with technologies and the way information is structured in various projects. Some tools have a steep learning curve that hinders students' effective participation and contribution. Some students commented that:

"Some of the projects am interested in did not support bug tracking system...I don't know how to use CVS and before I learn it, the semester will be over...Some bug reporting document guidelines are too long and has many versions...I couldn't describe exactly some bugs...project has 2 websites and I don't know where to report my bug."

Furthermore, educators may face difficulties in recommending suitable projects to students. However, it is important to note that, many students preferred working in gaming and VoIP projects because these software domains represented what they called "more exciting stuff".

B. Teachers' Experience:

The ne(x)t generation of students learn everything we have to teach them. They start asking more and more questions for which many of us teachers may not have answers. Students use Web 2.0 and many other technology gadgets to take their study of computer science or software engineering to new levels, to the extent that our teaching is no longer viewed as interesting and challenging.

The validation of this teaching and learning model through pilots, the author have come to learn that FOSS projects and communities provide many opportunities for students to grow and develop while still in graduate school. The FOSS context also provides teachers with the opportunity to continue to take part in the growth of students, to learn and discover new software and tools, and network with other teachers and project communities.

However, teaching in an Open Source like manner requires that teachers spend endless hours (24/7) to facilitate the learning process; continuously and promptly responding to students' emails, monitoring projects activities, and sending materials or links to resources to students. Sometimes, teachers need to intervene and guide some students in their projects, and even deal with enquiries or comment received from other project participants. Some students are also anxious as to whether they will get better grades by participating in FOSS projects or what they need to do in order to score maximum points for their grades.

VI. Conclusions

FOSS projects can provide unique opportunities for computer science students to improve their engineering skills. Students can enhance their career prospects by continuing their FOSS involvement long after they have graduated. FOSS also presents computer science and engineering departments the opportunity to improve their pedagogies and expose their students to realistic software development environments. Even though Open Source skills are as valuable as proprietary software experience and FOSS might not be a replacement for formal computer science education and training. The teaching and learning model presented in this paper may give computer science educators another pedagogical insight which may improve the way they can make computer science teaching and learning more relevant and meaningful for their students.

Acknowledgment

The author wish to thank the anonymous reviewers of the 3rd International Conference on Computer Technology and Development (ICCTD2011) for their comments and suggestions on the early version of this paper. Special thanks to the Software Engineering Group (SWEng) of the Department of Informatics, Aristotle University, Greece, where the pilot studies were conducted.

References

[1] M. Joanne, *et al.* "Software Engineering 2004: ACM/IEEE-CS guidelines for undergraduate programs in Software Engineering," in *Proc. 27th Int'l Conf. on Software engineering*, New York, 2005, pp. 623-624.

[2] K. Toth, "Experiences with open source software engineering tools," *IEEE Software*, vol. 23, no. 6, pp. 44-52, November 2006.

[3] E. Heidi, M. Ralph, L. Trishan, and H. Gregory, "Holistic Software Engineering Education Based on a Humanitarian Open Source Project," in *Proc. 20th Conf. on Software Engineering Education & Training*, Washington DC, USA, 2007, pp. 327-335.

[4] Jaccheri, L. and Osterlie, T. Open Source Software: A Source of Possibilities for Software Engineering Education and Empirical Software Engineering. *1st International Workshop on Emerging Trends in FLOSS Research and Development*, 2007, pp: 5-5.

[5] S. Sowe, "A Model for Teaching and Learning in Free and Open Source Software Projects: A Constructivist Approach," in *Proc. 3rd International Conference on Computer Technology and Development*, vol. 2, Chengdu, China, 2011, pp. 525-530.

[6] S. Sowe and I. Stamelos, "Involving Software Engineering Students in Open Source Software Projects: Experiences from a Pilot Study," *Journal of Information Systems Education (JISE)*, vol. 18, no. 4, pp. 425-435, December, 2008.

[7] I. Stamelos, "Teaching software engineering with Free/Libre Open Source Projects," *International Journal of Open Source Software & Process*, vol. 1, pp. 72-90, May 2009.

[8] J. Long, "Open Source Software Development Experiences on the Students' Resumes: Do They Count? - Insights from the Employers' Perspectives," *Journal of Information Technology Education*, vol. 8, pp. 229-242, May 2009.

[9] A. Meneely, L. Williams, and F. Gehringer, "ROSE: a repository of education-friendly open-source projects," *SIGCSE bull*., vol. 40, no. 3, pp.7-11, June 2008.
[10] S. Sowe, L. Angelis, and I. Stamelos, "A Framework for Teaching Software Testing using FOSS Methodology. Open Source Systems," IFIP *International Federation for Information Processing*, vol. 203, pp. 261-266, June 2006.
[11] L. Moreno, C. Gonzales, I. Castilla, E. Gonzalez, and J. Sigut, "Applying a constructivist and collaborative methodological approach in engineering education," *Comp. & Education*, vol. 49, no. 3, pp. 891-915, November, 2007.
[12] E. Wenger, *Communities of practice: Learning, Meaning, and Identity*. Cambridge University Press, Cambridge, 1998, ch. 6.
[13] R. Luckin, "The learner centric ecology of resources: A framework for using technology to scaffold learning," *Computers & Education*, vol. 50, no. 2, pp. 449-462, February, 2008.
[14] E. S. Raymond and R. Moen. (Rev 3.4, 29 September, 2006). How to Ask Questions The Smart Way. Available: http://catb.org/~esr/faqs/smart-questions.html, Tuesday, December 20, 2011.

Sulayman K. Sowe from The Gambia holds a PhD in computer science from Aristotle University, Greece, 2007; a Masters degree and Advanced Diploma in computer science from Sichuan University, China, 1995 and 1997, respectively; a Bachelor degree in Science Education, specializing in Physics from Bristol University, UK, 1991; and a Higher Teachers Certificate in Physics and Chemistry from Gambia College, The Gambia, 1989. His research interests includes Free and Open Source software development, Knowledge Management and Knowledge Sharing in distributed software development environments, Social Network Analysis, Information Systems Evaluation, Software Engineering Education, Digital Divide, and Information and Communication Technology for Development (ICT4D).

He is currently pursuing a Postdoctoral Research at the United Nations University Institute of Advanced Studies (UNU-IAS) in Yokohama, and he is a Visiting Scholar at the National Graduate Institute for Policy Studies (GRIPS) in Tokyo, Japan. He worked as a Senior Researcher at UNU-MERIT, Maastricht, Netherlands; as a Research Fellow at the Department of Informatics, Aristotle University, Greece; as a Database Manager at the Medical Research Council, The Gambia; as an Operations Officer, Assistant Registrar II, and Systems Administrator at West African Examinations Council; Director of Information Technology, Human Resource Development at the Ministry of Education, The Gambia; as a Teacher and Head of Science Department at Muslim High School, The Gambia. He holds various fellowships including the Japan Society for the promotion of Science (2010-2012), the Greek Scholarship Foundation (2003-2007), and the UK Overseas Development Agency (1989-1991). He as over fifty publications in various journals and has delivered keynote speeches in his specialization in various international conferences and seminars. He is the co-editor of two books: "*Emerging Free and Open Source Software Practices"*. IGI Global Publishing, 2008 and "*Free and Open Source Software and Technology for Sustainable Development*". UNU Press (2012, forthcoming).

Dr. Sowe is a member of a number of international, program committees and editorial review boards, including the IGI Global Editorial Advisory Review Board, the "International Journal of Open Source Software & Processes" (IJOSSP).