PROCEEDING

International Conference on *Open Source* for Higher Education

The Role of Open Source Software and Open Education Resource in Improving the Quality of Higher Education

March 15th, 2010
8.00 AM - 05.00 PM
Senate Hearing Room, Rectorate Building
SEBELAS MARET UNIVERSITY (UNS)
SOLO - INDONESIA

Organized By :

ISBN: 979-498-560-0

9794985600

CONTENTS

A Proposed Framework for an Ideal Information Technology / Computer Science Education

Andi Wahju Rahardjo Emanuel
Informatics Bachelor Program
Faculty of Information Technology
Maranatha Christian University
Jl. Suria Sumantri no. 65, Bandung, Indonesia
awreman@gmail.com

Yenni M. Djajalaksana
Information System Bachelor Program
Faculty of Information Technology
Maranatha Christian University
Jl. Suria Sumantri no. 65, Bandung, Indonesia
ymd261@yahoo.com

ABSTRACT

The use of Open Source Software (OSS) has grown rapidly within the past decade. Despite the increased importance of OSS in the Information Technology / Computer Science (IT/CS) field, the OSS has not been sufficiently addressed in the IT/CS education. At this time, the IT/CS curriculum still employs mainly proprietary Software Engineering (SE) methodologies. This discrepancy for OSS skills acquisition creates a knowledge gap in our IT/CS graduates. The gap will reduce our IT/CS graduates current and future employability if they do not acquire the necessary skills to develop and implement OSS. To anticipate this challenge, the authors propose a framework to combine the advantages and minimize the negative aspects of both OSS and SE environments. The main components of our proposed framework includes proper introduction to basic OSS skills, formation of diverse skills groups, system development using modularity principles, and using the SE methodology for monitoring.

Keywords

Open Source Software, Software Engineering, education framework, Computer Science, Information Technology

1. INTRODUCTION

OSS is a growing trend in the world of IT/CS. While the academics view OSS as a trivial movement in the early 1990s, it has become a major alternative software development methodology to the existing proprietary/commercial Software Engineering methodology. The OSS movement was first initiated by Eric Steven Raymond in his paper "Cathedral and Bazaar" with his 19 lessons [1] and by Richard Stallman in his paper "Why Software Should be Free" [2]. Slowly, the seemingly chaotic and "bazaar-like" movement in software development gains greater attention. Several examples of the first high quality applications from this movement are the Apache Web Server, Linux Operating System, Firefox Web Browser, OpenOffice productivity suite, and many more.

Unfortunately, the OSS acceptance in the higher education's IT/CS curriculum is not flourishing at the same pace with the growing popularity of OSS. The IT/CS curriculum, such as one recommendation by the ACM (Association of Computing Machinery[3], still focuses on the modern SE methodology. This SE methodology is mostly applicable only to the development of in-house / proprietary software. The fact that the OSS receives rising acceptance in the software development field is not adequately acknowledged in the formal IT/CS curriculum. As a result, this creates a knowledge gap in the IT/CS graduates. If the curriculum does not attempt to catch up with the development of OSS, this gap will be much wider. In order to close this gap, we believe a framework that combines the best concepts from both the OSS and SE methodologies should provide an ideal framework for IT/CS education.

2. OSS VERSUS SE ENVIRONMENTS

Both OSS and SE methodologies are not perfect. Instead, both have their own advantages and negative aspects. The following sub-sections introduce the advantages and negative aspects of both methodologies.

2.1 Open Source Software (OSS)

OSS is an alternative software development methodology based on several distinct characteristics:

- The source code of the application is freely available for everybody to download, improve and modify [1]. There are no restrictions for anyone to download and examine the source code. However, expected contributions from the developers are expected for the improvement and modification of the source codes.
- Peoples who contribute to the development of the OSS project usually belong to a group known as an Open Source Community. The membership of this community is voluntary in nature. The main media of communication among these developers is the Internet, although other media might be used.
- The development process of the OSS projects is mostly ad hoc and is lacking the formal methodologies normally found in the modern SE methodology. The main activities of OSS projects are to built features and to fix bugs.

Throughout the development of OSS, the developers and communities depend on a number of collaboration tools. CVS (Concurrent Version System) is used as the repository and version control of the source codes. Bug detection database software such as Bugzilla or TRAC is employed to keep track of all the bugs during development. All the proposed functionalities and suggested improvements are kept in a TODO List / Wish List. The documentation of the software is maintained in a Wiki environment. The communication among the developers and the communities are done electronically via websites, forum, mailing list, etc. The distribution and modification of the software are controlled by specific licensing scheme such as GPL, LGPL, etc [4]. Currently there are many portals available to host and to assist OSS developers initiating and managing their projects. Several samples of the more popular portals are Sourceforge (http://sourceforge.net), Google Code, Freshmeat, etc.

OSS provides numerous benefits that are not found in the modern Software Engineering methodology. This benefits show the true merits of the system [5]. The modularity and option values are the two main points that contribute to the success of OSS [5][6]. The modularity of the architecture will lower the barrier of newly involved developers to participate the project [5]. The option value will allow developers to innovate without impacting the entire project [5].

OSS methodology also has some disturbing negative aspects. While the main activities of the OSS methodology are to build new features and to fix bugs, there are few formal processes being conducted. The main decision that determines the faith of an OSS development project is usually in the hand of a small number of core developers using "benevolent dictator" principle [6]. Moreover, the commonly found management processes in modern Software Engineering methodologies are missing such as the requirement management, project management, metrics estimation, scheduling, and test suite design [7]. The other major negative aspect is the lack of documentations because documenting is not considered important by most OSS developers.

2.2 Software Engineering

IEEE (Institute of Electrical and Electronics Engineers) defines SE methodology as the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software [8]. SE methodology consists of several phases of development that are performed in a single run or iterative manner:

- Requirement Management: collecting, analyzing and managing requirements of the software being developed.
- Software Analysis, Modeling and Design: conducting technical analysis of the software, continued by modeling and designing a software based on the identified requirements.
- Software Coding: developing the source codes based on the design. The software may be developed in a procedural or object-oriented fashion.
- Software Testing and Acceptance: defining prosedures, performing test and setting acceptance criteria for the software.

These phases are performed in a series of steps such as waterfall method, spiral model, etc. In a more advanced Software Engineering, more anticipations on the changing requirements and environments are introduced such as with Extreme Programming, Rapid Prototyping, etc.

SE methodology provides many advantages during the software development phases. There are a number of studies and theories developed based on this methodology such as project estimation (COCOMO, function points, etc.), software metrics, cost estimation. Many techniques pertinent to the project execution, project control, task delegation, and task assignment are develop based on modern SE principles. One of the main advantages of this methodology is the documentation processes. These documentation processes are thorough and precise to follow the International standards.

Unfortunately, SE methodology also has some negative aspects. The SE principles assume that almost everything can be predicted and anticipated beforehand. Software engineering subject is different in comparison to many other subjects of engineering (such as mechanical, electrical, etc.). Its rapid and continuous advancements in the IT/CS discipline make it difficult to keep the same and stable environment. Thus, the continuously changing requirements will always affect the scope of the project. The Standish Group's Chaos report[9] shows that among many software projects, only a small number of projects is completed within the original planned budget, time, and scope. A bigger percentage is partially successful or failed. The previous assumptions in SE that everything is predictable and can be anticipated are hard to be fulfilled.

2.3 Observations

Teaching the software development concepts is a very challenging task for many IT/CS instructors because students often view the materials in the software development course as academic materials with no practical values [10]. At the same time, to involve students in a real software development experience using the proprietary SE methodology will require many aspects and collaborations with the clients who might not want to risk their projects by involving inexperienced students. The OSS opens up the opportunity for students to be in a real and practical environment to learn the software development concepts.

It is also important to recall that teaching solely based on SE methodology without considering the growing trend of OSS may create knowledge gaps in IT/CS students. Therefore, as an added value, the OSS methodology is suitable to give students the experience in a knowledge sharing environment while at the same time it will increase their coding skills in a much fluid and flexible environment. When students begin their early involvement in the Open Source Community, the added benefit will be the potential acknowledgment of their existence among other existing and future developers.

Learning solely based on an OSS methodology will not be the best way either. Numerous studies show that in the software development courses using OSS projects, students may face frustrations due to its ever changing environment [11] and they face a steep initial learning curve [10] to learn the OSS programming skills. Many students also struggle to acquire the basic understanding of OSS in the beginning of their OSS experience [12]. From another perspective, the instructor who teaches software development methodologies using the OSS will face a higher challenge in their preparation [10]. Not only that the instructor must familiarize him/herself with the codes, but he/she also must keep up with the most recent development in OSS movements. Instructor must also prepare the group project carefully to avoid project failure. This involves instructor-led project group formations. In a course that utilizes OSS project for students' learning, self-formed groups of students are those who fail to deliver the course project assignment. This was due to the lack of diverse skills in the team [13]. Therefore, prior to using OSS for teaching software development concepts, instructor's role in the preparation of an OSS project will be more complex and challenging.

Considering the benefits and the negative aspects of both OSS and modern SE methodologies, it is possible to create a middle ground that combines the benefits of both approaches while minimizing their negative aspects. While maintaining the course main materials of the SE principles, OSS provides students with a "world-size laboratory" and "world-wide staff" [14] due to the nature of open contributions from any developers all around the world. While working on an OSS project, students also contribute their work to the international community, while their projects will remain available for future improvement [14]. By involving students in a large-scale collaborative software development project where students are encouraged and demanded to study independently and collaboratively, OSS projects will improve students' self-studying capability, information searching capability, and team work skills [13].

Evidence on the effectiveness of such OSS projects for university students has been investigated in a number of studies. For example, in a controlled experimental study by Park and Jensen [12], the researchers investigate the effectiveness of the visualization tools to teach programming concepts using an open source project. The visualization tools help to provide a more efficient and effective way to understand the OSS codes by shortening their learning curve [12] in acquiring OSS basic programming skills. In another study, Pedroni, et al. [15] conduct an experiment where students receive an open source project assignment in programming. While students are struggling at first with the fear of failing the assignment, the OSS project is fruitful. At the end of the project, students are satisfied for the achievement and feel accomplished [15].

3. A PROPOSED FRAMEWORK FOR IT/CS EDUCATION

A combination of both the mature SE methodology for proprietary/ commercial software development and OSS methodology should provide the ideal education environment for IT / CS students. The power of the community in the OSS methodology should be combined with the power of planning, control and documentation from the SE methodology. A proposed framework to combine the advantages and to avoid negative aspects of both methodologies is presented below.

3.1 Preparation

In order to achieve the intended purpose the IT/CS education based on the combination of OSS and SE, there is a need to familiarize the IT/CS students to both environments prior to its implementation. Several examples of the preparations are:

- Specific preparation to acquire the OSS basic programming skills of the standard software coding practice prior to the OSS projects assignment (to avoid spending too much times on teaching the basic coding skills).
- Proper introduction to the world of OSS and its communities. This approach will allow the teams of student developers to learn the best coding practices from existing OSS projects.
- Proper introduction to the modern SE methodology. The focus of the introduction should be to the requirement management methodology, software analysis and design, and software testing techniques.

3.2 Community Building in IT / CS Education

It is important to prepare the group formation of the IT/CS students for the OSS project candidly to ensure its success. The students should be formed into groups with diverse skills of basic programming for developing OSS and the foundation of SE methodology skills. Each group will be formed as community-like clusters as shown in Figure 1, which consist of an administrator and one or more developers. The instructor should assign a single module of the software system to be developed by the group, and they should be supervised by an instructor who monitor and ensure knowledge sharing activities which are common in typical OSS communities. The group should communicate intensively within the group and among other groups in order to build the system. The result of the discussions and exchange of information should be documented properly using the Requirement Management steps of the SE methodology.

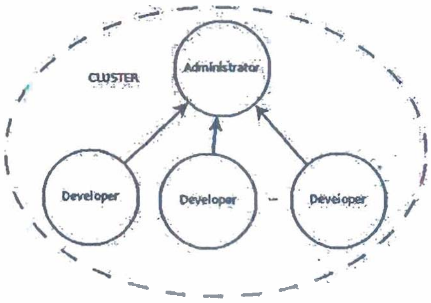

Figure 1. Structure of Single OSS Community Cluster

Even though the Requirement Management step is being used to record the process, it should be noted that the system requirements will continuously change. The system being developed by the community-like cluster will continuously evolve following the dynamics of the group itself or the changing software systems. The requirements should be continuously modified and revised to keep up with the evolution. The existence of Open Source documentation tool such as Wiki enables the immediate modification of the documentation to be published. The other developers then will be able to observe and follow any modifications.

3.3 System Development in IT / CS Education

Modularity has been identified as the key success factor of OSS. The modularity will ensure the system to evolve continuously in an evolutionary manner. Thus, the software system being developed should be arranged to be as modular as possible. The modularity of a software system is based on several factors [16]:

- Small modules: the size of each module should be as small as possible during in its development.
- Large number of modules: each module performs a single simple task. The larger the number of module, the more tasks and features are accomodated in the system.
- High complexity: the architectural complexity should be high to ensure the integrity of the system and the ability to accomodate new features to be added in the future by the

group.

- High cohesion: the structural integrity of the code in every module should be high to ensure high quality of the software system.
- High fan-in: one coupling parameter. The higher the fan-in (module is being called by many other modules), the higher the modularity of the system.
- Low fan-out: another coupling parameter. The lower the fan-out (module is calling another module), the higher the modularity of the system.

By following the modularity principle, the software system will consist of many modules communicating with each other. Each module should consist of several classes. The following Figure 2 shows the structure of a single module written in an object oriented programming language such as Java.

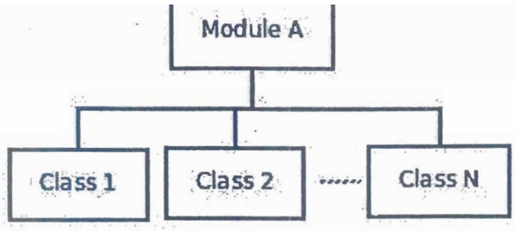

Figure 2. Typical Structure a Single Module

While developing modules of the software system with collaboration tools of OSS, the power of analysis and modeling existed in Software Engineering can be utilized to measure the parameters of the modules. Moreover, the modern testing techniques of Software Engineering methodology such as black box testing and white box testing should be used to cover the lack of proper testing techniques in the OSS methodology.

3.4 System and Group Co-Evolution

One of the OSS main characteristics that does not exist in a modern SE methodology is the co-evolution of the community and the system software. This should be adapted in the IT/CS education. A system software initially created as a single module as shown in Figure 2 with a cluster of developers as shown in Figure 1. A module may evolve horizontally in which the number of classes are increasing which correspond to the increasing capability of the module. More importantly, a module may evolve vertically in which a single class is evolving into a module as illustrated in Figure 3 [17].

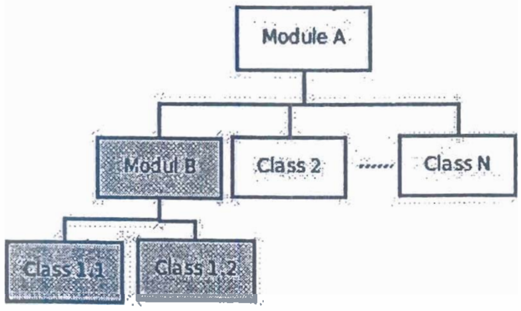

Figure 3. Vertical Expansion of Module

As shown in Figure 3, the formerly Class 1 evolves into Module B that has Class 1.1 and Class 1.2 as its members (shown in grey shade boxes in above figure). The system has evolved into a two-module software system. Module B is then has its own cluster of developers similar to Figure 1.

As the system is continuously evolving, the system becomes larger and more complex architecturally with many features being added over time. The structure of the community clusters is growing, corresponding to the increased number of modules. The following Figure 4 is an example of hypothetical community structure showing many clusters.

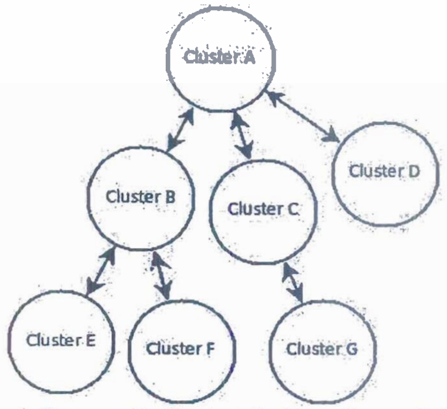

Figure 4. Community Cluster of a Open Source Software

4. CONCLUSION

OSS is a new mainstream in software development methodology. Even though this methodology is gaining an acceptance from the software developers community, the integration of the OSS methodology into the IT/CS curriculum is still lacking. Most students in IT/CS are still taught with the modern Software Engineering methodology. This condition will certainly create a knowledge gap in the IT/CS graduates. Both OSS and Software Engineering methodologies have their own benefits and negative aspects. However, when they are combined into a single integrated framework, it can accommodate an ideal education environment for IT/CS students. This framework will focus on combining the benefits of both methodologies and eliminating their negative aspects. The adaptive nature of OSS combined with the guided Software Engineering methodology will provide the best combination for IT/CS education.

The proposed framework to combine the two software development methodologies is described. The main processes of the framework are:

- Introducing the IT/CS students about OSS and SE methodologies, accompanied with the necessary basic programming skills.
- Forming groups of students with diverse skills in both OSS and SE into community-like clusters similar to OSS Community structure.
- Building the system in evolutionary manner using the modularity principles. The system and the community will co-evolve.