Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 56 (2015) 459 – 464

Procedia Computer Science

International Workshop on Enterprise Web Application Dependability (EWAD 2015)

Open Source Software (OSS) Quality Assurance: A Survey Paper

Salem S. Bahamdain[a]

[a] Collage of Computer & Information Systems, Umm Al-Qura University, Makkah, Saudi Arabia, ssbahamdain@uqu.edu.sa

Abstract

Open source software (OSS) is a software product with the source code made public so that anyone can read, analyze, and change or improve the code. The use of this software is under a license, like Apache, GNU, MIT, Mozilla Public, and Eclipse Public License. Open source software development (OSSD) provides high quality assurance through user testing and peer reviews. The quality of these products depends on the size of the product community. This paper discusses the stakeholders of the OSS community, the quality assurance frameworks and models proposed in some studies, some statistics about OSS, the problems that affect the quality of OSSD, and the advantages and disadvantages of OSS compared to closed source software. This allows us to understand how we can achieve and improve the quality assurance and quality control of OSSD.

Peer-review under responsibility of the Conference Program Chairs

Keywords: OSS, Open Source; Quality Assurance; Open Source Development Model; OSSD

1. Introduction

Today, open source software (OSS) provides a lot of services and products for companies, industry, governments, and education organizations, including The White House, GM, CNN, Stanford HC, BestBuy, and Umm Al-Qura University, as alternative products to closed source or commercial software products. There is a lot of successful OSS, like Apache, PHP, Nginx, MySQL, and MariaDB. OSSD is a kind of distributed software development base using the peer-review technique, and the development team is distributed across the world in different time zones.

* Corresponding author. Tel.: +966-5600-77929 .
E-mail address: ssbahamdain@uqu.edu.sa

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2015.07.236

This increases the difficulty of achieving quality assurance (QA), as do the risky development practices in open source software development (OSSD), such as unclear requirements elicitation, the ad hoc development process, the little attention paid to quality assurance and documentation, and poor project and quality management[1]. The development must consider much more than writing the code somehow. Every organization looks for very good architecture; reliable, testable, and maintainable code; and methodologies to support and maintain their software. To achieve these, software quality assurance and standards play an important role. Due to the fact that OSS is publicly available, this will improve the quality in three aspects:

- Service level
- Productivity
- End-user satisfaction

Also, OSS is more flexible in term of productivity and increases motivation and performance; it allows faster bug detection and error resolution compared to closed source software[2].
This paper covers the concepts of OSS and QA. In Section 2, we look at the concept of OSS, the problems in OSSD, and the onion model of stakeholders. In Section 3, we review OSS application domains and their sizes. In Section 4, we consider QA in OSSD. In Section 5, we analyse one of the frameworks for QA in OSSD[1, 2].

2. The Concept of OSS

There are many definitions for the concept of OSS, such as: "software which has ability to distribute freely with available source code through the internet and using unpaid people that can modify the code freely, is open source software"[3]. Some of the definitions identify the following characteristics[4]:

- Distributed software
- Free software
- Available source code
- Developers communicate through the internet
- Developers are users
- Unpaid and large amount volunteers

2.1. The Structure of Developers in OSS

There are four groups of OSS developers:

- Core developers
- Co-developers
- Bug reporters
- Users

- The core developers are a small group of expert developers who are responsible for the main core functionality of the system by writing high-quality code; managing and controlling the system; and making the plans, goals, and roadmap for the product.
- The contributing developers are a bigger group of developers who directly affect the software development. They have the ability to add new features (depending on code modularity) and to do some other tasks, like fixing bugs and peer-reviewing code[5].
- The bug reporters are responsible for testing the system. Some of this task can be done by the core developers and contributing developers, as well as by the users of the system. One of the main tasks for this group is to ensure that more people will test the system on many different platforms (if the system supports this) [6].
- The users utilize the system; some of them are the developers.
  Sustainable software development community groups can be described in a simple onion model, as shown in Fig.1[6].

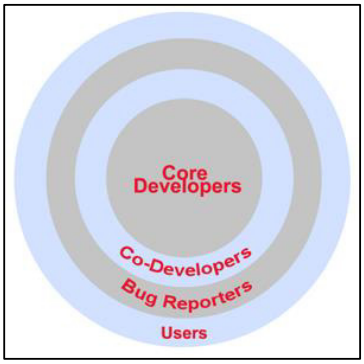

Fig. 1. The onion model of a sustainable software development community

2.2. The Pros and Cons of OSSD

There are many pros and cons of OSSD [2]. We will discuss these according to three aspects: the users, the developers, and the system itself. The pros for users include flexibility, strong value, code availability, possibility to modify the code, knowledge sharing through the community, and increased motivation.

For the developers, the pros include the ability to create customizable solutions, the potential to re-use many existing parts and functionalities, reduced production time, and increased motivation [7].

For the system, the benefits include faster bug detection and fixing, more reliability, the freedom to customize, cost-effectiveness, re-usability [8], rapid evolution, portability, and multitude licensing [9].

The cons for users include incomplete or bad documentation, unstructured development methodologies, and irresponsible individuals.

For the developers, the disadvantages are the lack of tools, collaborating with new developers, and reviewing large projects.

For the system, the cons are the lack of formal process-centralized management release and documentation, poor design, no single responsibility for problems, version proliferation, and the difficult estimation of manpower requirements, complex licenses, and high short-term cost.

3. OSS Application Domain and Size

There are many sites that host OSS applications (e.g. sourceforge, github, and bitbucket). These sites allow developers from anywhere in the world to access the code repositories and push, pull, and fork the code.

For example, there are 18 categories on sourceforge as application domains for OSS. The top five are internet applications (15.4%), software development (15.1%), systems (12.4%), communication (10%), and games/entertainment (9.3%). More than 70% of these projects are still in the early stages or are at the end of the development lifecycle[1]. One study categorized 100 projects listed on sourceforge and found that 86.2% of the projects had fewer than six developers; 1% of the projects had more than 16 developers, and these were large projects [10].

4. QA in OSSD

Software QA is defined as a “set of systematic activities providing evidence of the ability of the software process to produce a software product that is fit to use” [11, 12]. QA must be a part of each stage in any software development activity or lifecycle.

The QA in OSS needs to understand the sustainable community, code modularity, project management, documentation, and test process management. To achieve high-quality software, OSS practitioners must fully understand these areas and how they are related to one another.

High-quality OSS depends on having a large sustainable community to develop code rapidly, to identify bugs efficiently, to debug the code effectively, and to build new features.

There are many key differences between OSSD and closed source software development in terms of quality management, as follows [6]:

- OSSD does not have a defined or documented development methodology, has weak project documentation, has unstructured and informal testing and QA methodologies, has no formal risk assessment process, and has few documented or measurable goals. Moreover, programmers define the requirements, defect discovery from black box testing happens at a late stage of development, empirical evidence regarding quality is not collected, team members choose to work as volunteers, projects often go straight to programming without planning and making a roadmap for the project or development lifecycle, and there is little project planning and scheduling.

- Closed source software development has well-documented and defined development methodologies, complete project documentation (and maybe a user guide or article/video tutorial), formal and structured testing and QA methodologies, analyst-defined requirements and designs, formal risk assessment processes and monitoring throughout all project phases, defined and measurable goals for the project, and defect discovery from black box testing as early as possible. Additionally, empirical evidence regarding quality is used routinely to aid decision making, team members are assigned to work with paid salaries, a formal design phase is carried out and signed off before programming starts, and lots of effort is put into project planning and scheduling.

5. OSSD Framework and Processes

Many studies have proposed frameworks to manage QA under OSSD. We will discuss one of them[1], including the QA aspects of defect detection, managing code changes effectively, collecting feedback from users, collecting and verifying bug reports, and developing new features.

The quality of OSS is the result of interrelated QA within the developer and user communities. This QA framework in OSSD is based on the typical processes performed by different project participants during the bug or defect detection stage, and it defines defects as the errors, flaws, mistakes, failures, or faults in the software that prevent it from behaving as intended. Defects can be reported by the developer or user communities; these reports must be verified or validated before the project is progressed to the development phase. The framework diagram is show in Fig.2[1].

In this framework, there are three areas:

- Process 1: Defect Detection
- Process 2: Defect Verification
- Process 3: Solution Verification

We will describe each of these areas.

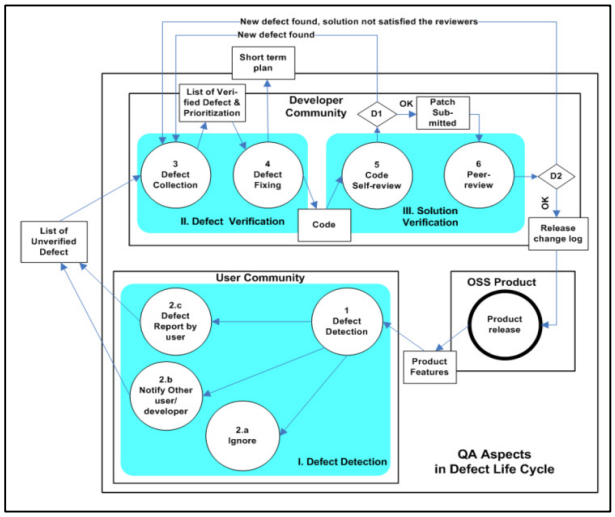

Fig.2: Framework for QA in an OSS project [1]

5.1 Process 1: Defect Detection

This process of detection and reporting provides information about the defects found by users or even developers through using the system and facing the errors or failures, so the process of finding defects in OSSD is like black box testing. When the user finds a defect, they will perform one of three actions: a) ignore that defect or error and continue using the software without reporting it, b) fill in a form for reporting the issue in a bug tracker system and describe what happened, or c) notify other users or developers about the defect by asking for their opinions about it through a mailing list or discussion forum (this is the most common result). More-experienced developers will perform simple analysis of the defect, verify it, and then add it to the bug tracker. Unverified defects will be added to a list until the next process.

5.2 Process 2: Defect Verification

This process for verification consists of defect collection and defect fixing/correction. After collecting a list of defects, they need to be verified as to whether they are real defects or false reports. One or more developers will read the issue report, review the code, make a comment or ask for extra information if needed, and then flag the defect as having one of four statuses: open, in progress, duplicate, or invalid. After that, verified defects must be moved to a list of verified defects. The next developer starts working on this list one by one and for each defect defines a defect fix. This is a short-term plan for fixing the issue; the developer who makes this plan is the developer who takes ownership of the defect. Code is then implemented to fix the issue.

5.3 Process 3: Solution Verification

This process focuses on solution verification or solution code review. While developing a solution for the defect, the developer continuously reviews and tests the code's functionality in what is called "self-code review". After this review, the developer will decide to push or submit these changes to the community if it is clear that the defect has been addressed. However, if there is still a defect or the new code or solution creates a new defect, the developer will report it through the bug tracker and restart process 2: defect verification.

If the developer chooses to submit a new solution to the community, the other developers or committers will review the code in what is called "code peer-review" and will check if the new code addresses the defect. They will then approve this new code if it meets the specifications; they will reject it if it does not meet the specifications or if it generates new defects. In the latter case, they will return the defect to the bug tracker, changing the status to something like "re-open" or "new", and will then start the process again from process 2, continuing like this until the defect is closed with a verified solution.

6. Conclusion

OSS is used in many critical applications in a lot of organizations and governments, so it is very important that there is high QA in OSSD. In this paper, we surveyed many studies on this topic. As discussed, there is a lot of work involved in OSSD, from planning and developing a clear roadmap to configuration management and project management. These aspects are needed to ensure the development of good-quality software because there are many more difficulties involved in OSSD than in closed source software development, such as having team members distributed around the world in different time zones and with different experience. Nonetheless, there is a lot of very good OSS with high QA. However, there are many OSS developers who do not care about the quality of their software or about performing basic tasks like black box testing or seeking users' feedback.

The topic of QA in OSSD requires further investigation because every day this software is becoming used in increasingly critical applications, so ensuring it is of high quality is essential.

References

1. Dindin W, Alexander S, Dietmar W and Stefan B. *Aspects of software quality assurance in open source software projects: two case studies from Apache project*. 33rd EUROMICRO Conference on Software Engineering and Advanced Applications. Lubeck; 28-31 Aug 2007. p. 229 – 236.
2. Atieh K and Riza S. *The process of quality assurance under open source software development*. IEEE Symposium on Computers & Informatics. 20-23 March 2011. p. 548-552
3. Gacek C and Arief B. *The many meanings of open source.*IEEE. Jan-Feb 2004. p. 34-40
4. Cubranic D and Booth K.S. *Coordinating open-source software development*. IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. Stanford CA; 16-18 Jun 1999. p. 61-66
5. J. J. Heiss. *The meanings and motivations of open-source communities*. Aug 2007, from Oracle, *http://www.oracle.com/technetwork/articles/java/opensource-phipps-137190.html*
6. Mark A. *Achieving Quality in Open Source Software.* IEEE.Jan-Feb 2007. p. 58-64
7. Xiong C.J et al. *A model of open source software maintenance activities*. IEEE International Conference on Industrial Engineering and Engineering Management. Hong Kong; 8-11 Dec 2009. p.267-271
8. Pekka A, Janne M, and Eila O. *Model-driven open source software development – the open models approach.* International Conference on Software Engineering Advances (ICSEA). Portugal; 20-25 Sep 2009. p.185-190
9. Abdullah R et al. *The challenges of open source software development with collaborative environment.* International Conference on Computer Technology and Development (ICCTD). Kota Kinabalu; 13-15 Nov. 2009. p. 251 - 255
10. Stefano C, Fabio M and Maria P. *From planning to mature: on the determinants of open source take-off.* July 2005, Marco Fanno
11. G. G. Schulmeyer & J. I. McManus, Handbook of software quality assurance 4th edition
12. Otte T, Moreton R and Knoell D. Applied quality assurance methods under the open source development model. Computer Software and Applications IEEE. Turku; July 28 –Aug 1 2008. p. 1247 - 1252