David Krovich

Dr. Dale Dzielski

SENG 581

14 March 2016

Quality Assurance Practices in the Open Source Software Development Model

The Open Source Software Development Model (OSSDM) is a software development model that has produced a large number of quality software projects such as the Apache Web Server, Mozilla Firefox Web Browser, and MySQL database. Often Open Source Software (OSS) has been developed outside of standard quality assurance practices for software development. This leads to the question what quality assurance practices exist in OSSDM to help ensure software quality? This paper will explore that question.

No discussion of OSS would be complete without a mention of Richard Stallman, the Free Software Foundation, and the GNU Project. Richard Stallman started the Free Software Foundation and the GNU Project with the goal to produce a completely free operating system. This operating system would be named GNU. GNU is a recursive acronym that stands for "GNU is Not Unix." The GNU operating system would be produced such that all components, source code, etc., would all be licensed under a free software license. This free software license became known as the GPL, or GNU Public License. The GPL is designed to have 4 basic freedoms associated with the software. The first freedom is "The freedom to run the program as you wish, for any purpose" (Free Software Foundation). The second freedom is "The freedom to study how the program works, and change it so it does your computing as you wish" (Free Software Foundation). The third freedom is "The freedom to redistribute copies so you can help your neighbor" (Free Software Foundation). The fourth freedom is "The freedom to distribute copies of your modified versions to others" (Free Software Foundation).

The GNU Project had a number of successful software projects. Two of the most famous are the GNU C Compiler (gcc) and a text editor/integrated development environment named emacs. Both gcc and Emacs are 100% free software and licensed under the GPL. They were to be the building blocks used to build the kernel of the GNU operating system. In an ironic twist of fate a programmer by the name of Linus Torvalds used emacs and gcc to build what became known as the Linux Operating System. Linus Torvalds was much more of a pragmatist and did not care about the more idealistic goals of Richard Stallman and the Free Software Foundation. This has lead to tenuous relationship between the Free Software Foundation and the Linux community over the years.

This divergence in ideas between the pragmatic benefits of free software versus the idealistic vision of free software supported by Richard Stallman and the Free Software Foundation effectively lead to the creation of the term Open Source Software. Free software is effectively a political movement with the goal of ensuring freedom of humanity by keeping all of the software that we use in our daily lives free as well. By contrast, open source cares much more about the pragmatic benefits of developing software in a more open environment and is much more business friendly.

There are many different licenses that are considered compatible with open source. Some of the more well known are the GNU Public License, Apache License, BSD License, and the

MIT License. The important point about all these licenses is that they are effectively copyright licenses. A copyright license dictates what happens when software changes from one hand to another or from one organization to another. Typically these licenses allow for access to source code and give the ability to redistribute software at no cost.

Software developed under OSSDM typically will have the following characteristics. The first is that the software will be developed under an OSS compatible license. This is usually selected very early in the development process and can help shape the user and development community. It is also very common for OSS to be developed by people in different physical locations with all communications happening via the internet. Mailing lists, wikis, source code repositories, online bug tracking systems, irc, etc., are all resources used in support of the OSSDM. OSS Projects also typically have similar hierarchies to support development. At the center are the Core Developers. These are the people in charge of the project and set standards and decide which code and features will be worked on. Outside of the Core Developers are Co-Developers. The Co-Developers typically work closely with the Core Developers but are not as central to the project. They may contribute small bug fixes or patches but most likely would not be responsible for an entire module or feature. The next level is Bug Reporters. Bug Reporters work on accurately reporting bugs using whatever bug reporting system or standard is used by the project. The last group is typically the end users. They use the software and often contribute via mailing lists or other discussion formats but typically don’t get involved in development or formal bug reporting.

OSSDM has some advantages with respect to software development. One of which is that developers are often users. One of the biggest complexities in software development is trying to communicate requirements from users to developers. If the developers are also users than this obviously makes understanding requirements a lot easier. Another advantage is with respect to the large communities of interested stakeholders that form around successful open source software projects. Having a community behind a piece of software provides necessary resources to a project such as testing, bug reporting, feature enhancements, etc. The last advantage I will mention is with the open source software toolchain. Over the years a lot of open source software has centered around GNU/Linux and the GNU/Linux toolchain. There is a wealth of quality tools written to support open source software development that are also open source. Having a vibrant selection of tools and a large community behind them is an important factor in the success of open source.

OSSDM also has some disadvantages with respect to software development. Open Source projects often suffer from poor documentation practices. Requirements documentation, requirements specifications, etc are often lacking in open source projects. Some of this can be attributed to the fact that developers are often users. If the developers already understand the requirements they may not feel as pressing of a need to document them. Another weakness is that open source software projects often have very poor design practices. Many open source projects start with a person or small group of people that will dive right into a project and immediately start coding without much thought about design.

"Quality Assurance (QA) is a systematic methodology that verifies whether products reach or exceed customer expectations or not." (Khanjani and Sulaiman 1) While often different from more traditional software development practices there are a number of quality assurance activities that do exist in the open source software world that help with software quality. One of the most important factors that can affect the quality of a project is having a large sustainable community behind it. Without a vibrant community behind it an open source software project can not be successful. The better the community the better the ability to develop code, find bugs, and design new features and functionality. According to many studies creating a sustainable community should be an Open Source Software Project's key objective. (Aberdour 59)

Getting accurate requirements is a huge problem with respect to producing quality software. As it was mentioned earlier in OSSDM developers are often users. This makes the quality assurance related to requirements extremely effective. It would be hard to design a better system than one where developers are users with respect to gathering and understanding requirements.

Defect detection is an important part of the quality assurance process. Most defect detection in OSSDM is similar to black box testing. Users running the software notice a particular defect or error in the operation of the software. The next step in defect detection is to report the defect. This depends a lot on the user. The user may ignore the issue and not report, may ask for advice on a mailing list or discussion forum, or if they are a little more experienced go and post a bug directly into the project's issue tracking system with some early defect analysis. There is no real barrier to entry to reporting a defect in an open source project. Typically most mailing lists, discussion forums, bug tracking systems, etc are open to the world in a transparent fashion. If an open source project has a vibrant community, then it will have a vibrant community of people hunting for bug defects which should lead to quality software.

After defects are detected and reported the next step in a quality assurance process is defect verification. Defect verification deals with determining if a reported defect is a valid defect. Defect verification usually occurs within the developer community itself but is also open for other users to comment and validate the existence of specific defects. The next step is to classify the defect as valid or invalid and assign a severity level. Once the defect is verified the next step is for a developer to take ownership for fixing the bug. The developer will typically communicate via a mailing list or bug tracking system that they intend to correct the defect. They then work on a solution to correct the defect.

After a defect is verified and a potential solution has been provided, the solution itself must be verified. The first step in OSSDM is for the developer to do a self review of the fix. After this self review the developer will then release a proposed fix for additional peer review. This can be discussed on a mailing list or within a bug tracking system. This creates a peer review process that helps assure software quality. Not only will the fix be reviewed in terms of fixing the desired defect, but it it will also be reviewed in terms of not introducing new defects. After going through peer review typically a fix will still need to make it through a gatekeeper to have the fix applied to the actual codebase. This provides one more level of quality assurance as another layer of code review will happen to all proposed solutions to verified defects. Once the code is merged and released the process resets itself and the new codebase is used by the community and new defects are found.

Most OSS projects use some sort of version control and typically the entire development process is done in a transparent fashion. Everyone from core developers to end users can see all patches and changes being made to the code base. Information from issue tracking systems or mailing lists are also available. Having all this information available in a transparent fashion provides for a constant peer review mechanism. All code is being constantly peer reviewed at all times. In addition to constant peer review there is also the hierarchy that institutes a peer review process and helps quality control. Changes have to evolve through all levels of the hierarchy from the user reporting a defect, to a bug reporter verifying the defect, to a co-developer providing a solution, to a core developer finally merging the new code.

In traditional practices testing typically constitutes one of the last steps performed to ensure a product meets user requirements and quality specifications. There is evidence of significant resources being spent on testing in OSSDM. One survey of open source projects show that 58% of projects spent more than 20% of time on testing, while more than 15% of the projects spent more than 40% of their time in testing (Zhao and Elbaum 69). A study of 13 software companies reported the average percent time spent in testing is 21% (Zhao and Elbaum 69). According to those numbers OSSDM appears to spend an adequate amount of time with respect to testing. It is interesting to note that the same survery had some interesting results with respect to code coverage. While respondents were familiar with coverage, only 5% of them employed tools to help assess it. Furthermore, a majority of the projects surveyed estimated less than 30% total coverage addressed by testing (Zhao and Elbaum 70).

Having good tools is an important asset towards overall quality assurance. It has been touched upon at earlier points in this paper but the OSSDM has produced a wealth of tools to aid in all facets of the development process. There are endless numbers of compilers, integrated development environments, development frameworks, testing engines, etc available to aid in the development of software. In many ways OSSDM is very developer centric and therefore it is no surprise that much attention has been spent by developers creating quality tools to aid in the creation of software.

Modularity is an another important aspect towards producing quality software. By designing components that are well defined and isolated from each other help ensure the desirable traits of low coupling and high cohesion among software components. Some of the most successful open source projects are ones that have developed a high level of modularity in the overall design of the components of the system. One study investigated 100 open source C applications and established a clear relationship between high modularity and quality. (Stamelos and Ioannis, et al.) This allows segments of the community to focus on certain parts of a projects and helps insulates changes to one component having unintended or adverse affects on another component. Modularity also allows a way to attract new developers towards a project. Rather than being overwhelmed trying to learn the entire system, by modularizing a system a developer can start by learning one small subcomponent of the system. It is an important aspect of OSSDM to be able to attract new developers and bug reporters so a modular design aids in the structure and culture of the community behind an open source project.

Another aspect that has an impact on software quality in OSSDM is the aspect of competition that exists within a software community. Within OSSDM there is the ability to add people and resources to a project. The open model allows people to join and contribute at their leisure. Often within open source communities there exists a competition among developers to find the best solution to a problem. This can lead to an interesting phenomenon where multiple people will suggest multiple solutions to a single problem. This has the advantage of evaluating multiple approaches to a problem and even seeing multiple implementations of a solution. The community can then evaluate each solution and ideally come to a consensus on which solution best fills the needs of the project. When coupled with modularity the competition inherent in a community can provide a very powerful quality assurance mechanism in OSSDM.

The OSSDM model uses some unconventional methods to develop software and even seems to contradict some well established software engineering principles. One famous principle is known as Brooks Law of Communication which states that communication overhead will increase exponentially as more people are added to a project. OSSDM supports large numbers of development communities with people scattered all over the globe. OSSDM has the ability to add new members fairly easily due to the tiered hierarchy structure that exists within typical communities. These projects do not seem to suffer from the communication overhead problem that is identified by Brook's Law of Communication. Another common well established software engineering principle is that the earlier bugs are found, the cheaper they are to fix. In traditional software engineering the idea is to flesh out potential bugs in the design level. In OSSDM bugs get fixed at code level rather than design level. (McConnell) Due to the typical development nature there is a constant peer review mechanism that is constantly evaluating each iterative step of development. This helps eliminate code level defects early in the development process.

While some quality assurance practices exist in OSSDM, there is much room for improvement. Some of the obvious areas in need of improvement are in the areas of documentation and design practices. In addition there needs to be a cultural change in the community to put more of an emphasis on code quality. There is an important aspect to getting it right the first time as opposed to having to go back later and repair or replace a faulty piece of code. This concept and way of thinking could have a dramatic improvement on the overall quality of open source software.

Another common weakness with open source software is the perception that it is software randomly thrown together with little concern for quality or testing. There is no standardized way to evaluate different open source software projects and compare the maturity level of practices within that project. Fortunately there is work in this area to come up with standards similar to CMMI but tailored specifically for OSSDM. One such model is the Open Source Compatibility Model and it provides a way to appraise projects and evaluate their maturity level. Having these kinds of standards could go a long way towards helping the adoption of open source software. This would allow projects to select only open source software that meets a certain level of standards in their process and practices in developing software.

Additionally there is also work being done to better define the processes and frameworks that can be used to perform quality assurance in OSSDM. One such framework being developed is the Quality Assurance Framework for Open Source Software (QAfOSS) (Otte, Moreton, and Knoell). This framework is a generic process framework that is separate from the development process framework and is geared specifically towards quality assurance practices in OSSDM. Having well defined frameworks such as QAfOSS will only help improve quality assurance practices in the OSS world.

In conclusion, while OSSDM differs from more traditional software methods it is clear that some quality assurance practices exist to help ensure software quality. The existence of these quality assurance practices help explain how projects like the Apache Web Server, Mozilla Firefox, and the MySQL Database all produce successful high quality software products.

Another point to note is the study of quality assurance practices within OSSDM is still a very new endeavor. More research needs to be done to study the current state of quality assurance within OSSDM in order to better define the current methods and practices in place. In addition to defining the current practices and standards there is plenty of room for improvement in quality assurance in OSSDM. Hopefully as more work and attention is brought to this area the open source communities will make quality assurance more of a higher priority in the development of open source software.

Bibliography

Bahamdain, Salem S. "Open Source Software (OSS) Quality Assurance: A Survey Paper." *Procedia Computer Science* 56 (2015): 459-64. Web.

Kuwata, Yoshitaka, Kentaro Takeda, and Hiroshi Miura. "A Study on Maturity Model of Open Source Software Community to Estimate the Quality of Products." *Procedia Computer Science* 35 (2014): 1711-7. Web.

Adewole Adewumi, Sanjay Misra, and Nicholas Omoregbe. “A Review of Models for Evaluating Quality in Open Source Software.” *IERI Procedia 4* (2013): 88-92.

Khanjani, Atieh, and Riza Sulaiman. "The Process of Quality Assurance Under Open Source Software Development".Web.

Otte, T., R. Moreton, and H. D. Knoell. "Development of a Quality Assurance Framework for the Open Source Development Model".Web.

Aberdour, Mark. "Achieving Quality in Open Source Software." *IEEE Software* 24.1 (2007): 58. Web.

Free Software Foundation. "What Is Free Software? - GNU Project - Free Software Foundation." *What Is Free Software? - GNU Project - Free Software Foundation*. N.p., n.d. Web. 06 Mar. 2016. <http://www.gnu.org/philosophy/free-sw.en.html>.

Wahyudin, D., et al. "Aspects of Software Quality Assurance in Open Source Software Projects: Two Case Studies from Apache Project".Web.

Otte, T., R. Moreton, and H. D. Knoell. "Applied Quality Assurance Methods Under the Open Source Development Model".Web.

Zhao, Luyin, and Sebastian Elbaum. "Quality Assurance Under the Open Source Development Model." *The Journal of Systems & Software* 66.1 (2003): 65-75. Web.

McConnell, Steve. "Open-Source Methodology: Ready for Prime Time?" *IEEE Software* 16.4 (1999): 6. Web.

Stamelos, Ioannis, et al. "Code Quality Analysis in Open Source Software Development." *Information Systems Journal* 12.1 (2002): 43-60. Web.