Reliability Issues in Open Source Software

R. K. Pandey
University Institute of Computer Science and Applications (UICSA)
R.D. University, Jabalpur (M.P.)
India

Vinay Tiwari
University Institute of Computer Science and Applications (UICSA)
R.D. University, Jabalpur (M.P.)
India

ABSTRACT
Open Source software in recent years has received great attention amongst software users. The success of the Linux Operating system, Apache web server, Mozilla web browser etc. demonstrates open source software development (OSS) as an alternative form of software development. Despite the phenomenal success of open source software the reliability of OSS is often questioned. The opponents of open source software claim that open source software is unreliable as the source code of OSS is available and the potential threats can easily be incorporated. Whereas the supporters claim OSS to be more reliable than proprietary software as the source code is open and freely available for scrutiny for all. This paper analyzes the reliability issues of open source software in contrast to the proprietary software. Various views of researchers on the reliability of OSS are studied and analyzed and a theoretical study is made to examine the reliability of OSS. Results of various surveys on reliability conducted by various researchers/agencies are also incorporated in support of reliability analysis of OSS.

General Terms
Design, Reliability, Security, Measurement

Keywords
Open Source Software (OSS), Software reliability, quality, proprietary software.

1. INTRODUCTION
In a recent survey conducted by Accenture [1] over 300 large blue chip organizations in both public and private sector of US and UK on the use of open source software, a very interesting result is found. "Two third of overall organizations (71%) increase using open source software not just for cost saving but for the improved reliability and better quality of open source software". Open source software during recent years has attracted software users who want high quality software but are not in a position to afford expensive commercial software. Software like Apache web server GNU Linux, MYSQL, Mozilla web browser, Open office, Perl programming language etc. have got phenomenal success among the software users. The amount of free and open source software increasing exponentially with the expansion of internet and currently there are at least hundreds of thousands of such software projects. Generically open source refers to a program in which the source code is available to the general public for use and/or modifications from its original design free of charge, whereas in the conventional commercial software the end product is in the form of binary. Open source software is having major impact on software and its production processes. Open Source Software developers have produced systems with functionality that is competitive with similar proprietary software developed by commercial software organizations. The success of open source software demonstrates the alternative form of software development processes. Software development is undergoing a major change from being a fully closed software development process towards a more community driven open source software development process [2]. The computer software development can now be broadly split into two development models [3]:

- Proprietary, or closed software owned by a company or individual in which 'binary' codes are made public and the source code is not usually made public.
- Open Source software (OSS), where the source code is released with the binary codes. Users and developers can be licensed to use and modify the code and to distribute any improvement they make.

Open source software development strategies are quite distinct from that of traditional software development methods. Free Open Source Software Development (FOSSD) is a way for building deploying and sustaining large software systems on a global basis and differs in many interesting ways from the principles and practices traditionally advocated for software engineering [4]. For example software design before the development and software testing before the release is hardly carried out in the OSS development. Secondly there is no single well defined development process for OSS and it can vary from project to project. The lack of unified development process and traditional software engineering practices raises question of reliability of OSS. Opponents of open source software claim that open source software are unreliable as the source code of OSS is available and the potential threats can easily be incorporated. On the other hand proponents of Open source software claim that the open source software are more reliable than closed proprietary software as thousands of independent programmers are involved in testing and fixing bugs of the software. This paper analyzes the reliability issues of open source software and examines the various views and claims of various researchers on reliability of open source software. Results of various surveys on reliability conducted by various researchers/agencies are also incorporated in the reliability analysis of Open Source Software.

2. WHAT IS SOFTWARE RELIABILITY AND ITS MEASURES
According to ANSI [5] software reliability is defined as "the probability of failure free software operation for a specified period of time in a specified environment". Informally reliability is defined as a measure of how closely a system matches its stated specifications. Although software reliability is defined as a probabilistic function and comes with the notion of time, it is noticeable that software reliability is different from traditional hardware reliability and it is not a direct function of time. In

hardware reliability with the heavy usage and as the time goes electronic and mechanical parts may become "old" add wear-out but software will not rust or wear-out during its life cycle. Software will not change over time unless intentionally changed or upgraded. Software reliability is based on the concept of failures. Failure of software occurs for variety of reasons like defects in code i.e. coding error, faulty software design, irrelevant input data etc. Reliability means the absence of defects which cause incorrect operation, data loss or sudden failures. It is obvious that software product having a large number of defects is unreliable. It is also clear that the reliability of the system improves if the number of defects in it is reduced. However, there is no simple relationship between the observed system reliability and the number of latent defects in the system [6]. For example, removing errors from parts of software which are rarely executed makes little difference to the perceived reliability of the product. Thus, reliability of a product depends not only on the number of latent errors but also on the exact location of the errors and on how frequently is the corresponding instruction executed. Apart from this, the manner of use of a product also determines its reliability. If it is selected input data to the system such that only the "correctly" implemented functions are executed, none of the errors will ever be exposed and the perceived reliability of the product will be high. On the other hand, if the input data is selected such that only those functions which contain errors are invoked, the perceived reliability of the system will be very low. So, the failures affecting different users differ in kind and frequency. Even software that performs well for most users may be ruinously unreliable for some of them.

Software reliability is comprised of three major mechanisms these are fault prevention, fault detection and removal and measurement to maximize reliability [7]. Measuring reliability means the measuring of defects. Identification and removing of software error contribute to increased reliability of software. Software reliability can not be directly measured, so other related factors are measured to estimate software reliability and compare it among products. Development process, faults and failures found are all factors related to software reliability.

3. OPEN SOURCE SOFTWARE AND ITS CHARACTERISTICS

Open source software is computer software that is available in source code form that permits users to study the software, to use software freely, change and improve software as per his requirements. Open source software is always released under a license that allows users to access, modify and redistribute the source code. In contrast, creators of proprietary software usually do not make their source code available to others. They provide only binary code, so users can run the software but cannot study, modify, or improve it. The OSS license gives users the following four essential 'freedoms' [8].

- to run the program for any purpose,
- to study the working of the program, and modify the program to suit specific needs,
- to redistribute copies of the program at no charge or for free, and
- to improve the program, and release the improved modified version

Feller and Fitzgerald [9] have outlined the following key conditions to define OSS:

1. The source code must be available to users.
2. The software must be redistributable.
3. The software must be modifiable and the creation of derivative works must be permitted.
4. The license must not discriminate against any user, group of users, or field of endeavor.
5. The license must apply to all parties to whom the software is distributed.
6. The license cannot restrict aggregations software.

Open source also encompasses a software development methodology. Open source software is generally developed on voluntary basis by the global network of developers. It is a collaborative software-development method that harnesses the power of peer review and transparency of process to develop code that is freely accessible. Open source draws on an ecosystem of thousands of developers and customers all over the world to drive innovation. The open source software development model has the following features [10].

- Collaborative, parallel development involving source code sharing and reuse
- Collaborative approach to problem solving through constant feedback and peer review
- Large pool of globally dispersed, highly talented, motivated professionals
- Extremely rapid release times

Open source development is described as a rapid evolutionary process, which leverages large scale peer review. The basic premise is that allowing source code to be freely modified and redistributed encourages collaborative development. The software can be incrementally improved and more easily tested, resulting in a highly reliable product.

4. DIFFERENCE BETWEEN TWO STYLES OF DEVELOPMENT

The Cathedral and the Bazaar [11] is the most frequently cited description of the open-source development methodology. In this book, Raymond makes the distinction between two kinds of software development. The first is the conventional closed source development. This kind of development methods are, according to Raymond, like the building of a cathedral; central planning, tight organization and one process from start to finish. The second is the progressive open source development, which is more like an "a great babbling bazaar of differing agendas and approaches out of which a coherent and stable system could seemingly emerge only by a succession of miracles." The Cathedral model represents the traditional commercial software development style, using small teams, tight management control, and long release intervals. The Bazaar model represents the style of releasing early often involving a large number of pool of developers working on the product.

According to an Apache case study [12] the usually mentioned main differences between commercial and open-source projects are:

- Open-source systems are built by potentially large numbers (i.e., hundreds or even thousands) of volunteers.
- Work is not assigned; people undertake the work they choose.
- There is no explicit system-level design, or even detailed design
- There is no project plan, schedule, or list of deliverables.

For commercial software engineers it might be surprising that open-source projects relying on far less design documents, contracts, project plans or development processes can have success.

Traditional software development starts with detailed requirements document that is used by the system architect to specify the system. The customer usually tenders a job within his monetary limits, with more or less detailed requirements description to service providers. The project manager has to make sure that the development costs run within the boundaries of the customer's budget. This in consideration, the whole project is scheduled into work units and iterations to consume the least resources possible while still fulfilling the requirements and making the deadline [13]. If costs run up to high the project fails. This means there is a lot of pressure on the development team during the project. At the same time they have to build it the way the customer wants it leaving the developers not much of a choice. On the other hand it is hard to run an open source project following a traditional software development method. In open source software development requirements are rarely gathered before the start of the project [14]. Instead someone comes up with an initial idea, usually because of some personal interest. With all important design decisions made and a presentable prototype, capable of raising expectations, it is time to go public and acquire fellows. Obviously this works best if project initiator has a good reputation already [11]. Now it is time to build the community, release early and often and live up to the bazaar style. In classical development methods processes are designed to produce quality. These processes are targeted to ensure that the costs run within their estimates that all requirements and deadlines are met and traceability is assured. This is why much time and effort is spent in requirements analysis, design and specification. In open-source development there are usually no hard deadlines to meet and requirements are not statued before hand. Therefore working units are not necessarily scheduled ahead of time. Instead developers tackle issues from their personal list of ideas, bug reports or change requests through community feedback as they appear.

5. RELIABILITY ISSUES IN OSS

The reliability is the important attribute of the software quality and it is generally considered that free is not better and the reliability of OSS is called into question by closed source proponents. They claim that the software in closed source is developed by the systematic approach and with strict guidelines of software engineering principles resulting a high quality and more reliable software. Whereas the open source violating the traditional software engineering principles, guideline and absence of the systematic plan of development resulting in a less reliable software. They further argue that since open source software is open, defects and security flaws are more easily found and this makes it easier for a malicious person to discover security flaws whereas closed source makes it more difficult for attackers to find and exploit flaws. They argue that 'if the software is in the public domain, then potential hackers have also had the opportunity to study the software closely to determine its vulnerabilities'. So the question arises "Is open source software more or less reliable than proprietary software?" The remaining discussion trying to find out the answer to this question on various theoretical as well as various analytical results.

In his most cited book Eric Raymond Says: "The general business case for open-source is reliability. Open-source software is peer-reviewed software; it is more reliable than closed, proprietary software. Mature open-source code is as reliable as software ever gets." Further he writes: "The core idea of open-source development is very simple: open-source programmers have learned that secrecy is the enemy of quality. The most effective way to achieve reliability in software is to publish its source code for active peer review by other programmers and by non-programmer domain experts in the software's application area". Peer review enables fellow software writers to ensure that the software will actually do what it is designed to do. Open source coding can be analyzed, audited, and vetted by dozens, hundreds, or even thousands of individuals who all expect to be able to use their software without problems. Several quantitative analyses are made to test the defect density of open source software. A study by Coverity [15] found that the Linux kernel had far fewer defects than the industry average. Code-analysis firm Coverity performed a four-year research effort and found that the Linux kernel has significantly fewer software bugs in it than the industry average. Coverity's approach reported 985 defects in the 5.7 million lines of code in that make up the Linux kernel. According to data from Carnegie Mellon University (CMU), a typical program of similar size would usually have more than 5,000 defects. The August 2005 study found an average of 0.16 defects/KSLOC, down from 0.17 defects/KSLOC, even though the amount of code had increased. Another study by Reasoning found that the MySQL database (a leading OSS/FS database) had fewer defects than a set of 200 proprietary programs used for comparison. In a similar manner to the previous study, on December 15, 2003, Reasoning announced its analysis results comparing MySQL with various proprietary programs. MySQL had found 21 software defects in 236,000 source lines of code (SLOC), producing a defect density of 0.09 defects/KSLOC. Using a set of 200 recent proprietary projects (totaling 35 million SLOC), the same tool found a defect rate of 0.57 defects /KSLOC, over six times the error rate. In open source model, code is written with more care and creativity, because developers are working only on things for which they have real passion. Participation of wider development community helped significantly in the defect repair. Another is that developers really care about reliability. Free software packages do not always compete commercially, but they still compete for a good reputation, and a program which is unsatisfactory will not achieve the popularity that developers hope for. What's more, an author who makes the source code available for all to see puts his reputation on the line, and had better make the software clean and clear, on pain of the community's disapproval.

6. SECURITY

Open source advocates argue that the open source model also means increased security, because code is in the public view, it will be exposed to extreme scrutiny, with problems being found and fixed instead of being kept secret until the wrong person discovers them. In January 1999, attackers were able to plant a Trojan Horse version of the TCP/Wrappers tool on a well-known

FTP site; since source code was available; the back door was quickly noticed and removed. In contrast to this with a monolithic operating system like Windows 2000, which has tens of millions of lines of secret, bug-ridden code. Without access to the source code, customers are 100% reliant on the good will and competence of the Microsoft Corporation, a reputation for self-serving behavior, and last but not least, it's a way that the little guys can get together and have a good chance at beating a monopoly. New Evans Data Survey [15] reports that Linux systems are relatively immune from attacks as security breaches are rare in Linux Environment. 78% of the respondents to the GNU/Linux developers' survey have never experienced an unwanted intrusion and 94% have operated virus-free.

7. BUGS TRACKING AND REMOVAL

Raymond states that "with enough eye balls, all bugs are shallow", which suggests that there exists a positive relationship between the number of people involved and bug numbers. In an open source environment bugs are discovered quickly and fixed within hours of their being detected. This is because the coding for open source software is open and transparent. Because people can look at it, they can easily figure out where the bug is. They not only discover bugs but also fix them and then report it to the maintainers as well as issuing an updated version of the software on their own authority. Research indicates that the open-source software - Linux - has a lower percentage of bugs than some commercial software. In a closed source development, only fewer people examining how the program works means that fewer bugs in the program will be found and fixed. On the other hand open source development environment has thousands of independent programmers testing and fixing bugs of the software resulting in a more reliable software.

8. OPENNESS BENEFITS

Since open source software is open, defects and security flaws are more easily found. Users have access to the human readable source code to the program. They have the option to consult the source code and fix the problem or at least tell the developers exactly where the problem is [16]. If the developers will not fix the problem, the user has the option of fixing it or hiring someone else to fix it. Users will no longer be at the mercy of one programmer or company which owns the source code and is in sole position to make changes [17]. Open source software will have more people looking at the program, and this usually means that more bugs will be caught.

9. DEVELOPMENT PROCESS

The OSS development approach has helped produce reliable, high quality software quickly and inexpensively. It is sometimes said that the open source development process may not be well defined and the stages in the development process, such as system testing and documentation may be ignored. However this is only true for small (mostly single programmer) projects. Larger, successful projects do define and enforce at least some rules as they need them to make the teamwork possible. In the most complex projects these rules may be as strict as reviewing even minor change by two independent developers.

10. SCHEDULING / COMMERCIAL PRESSURES

Free software are developed in accordance with purely technical requirements. It does not require to think about commercial pressure that often degrades the quality of the software. Commercial pressures make traditional software developers pay more attention to customers' requirements than to security requirements, since such features are somewhat invisible to the customer. The fixed schedule imposed on the software developers also degrades the quality of the software [18]. The proprietary software company probably announced that the software would be released on a particular date. This places an onus on the software developers, forcing them to release software that's possibly not ready simply because the schedule demands it. On the other hand open source projects are largely immune from "time-to-market" pressures. A system need not be released until the project owners are satisfied that the system is mature and stable.

11. CONCLUSIONS

In this paper study is made to find out the reliability of open source software in comparisons of proprietary software. Reliability is difficult to measure and problems reports are not necessarily a sign of poor reliability. However our study shows that open source software in many ways is quite equivalent or better than the proprietary software. This behavior is also supported by various quantitative analysis reported by various research agencies. These studies show that in many cases, using OSS programs is a reasonable or even superior approach compared to their proprietary competitor. The main factors that make open source software more reliable are the facts that developers are usually also users of the software, developers are members of a community of developers, public availability of the source code and fast bug removal practices since thousands of independent programmers testing and fixing bugs of the software. Widely used open source programs tend to be more reliable; if they were not reliable, they probably would not be so widely used.

12. REFERENCES

[1] Accenture, Aug 2010, available at http://www.accenture.com retrieved on 15.4/2011.

[2] Deshpande Amit, Richle Dirk (2008), The total growth of Open source, Proceedings of the fourth conference on Open Source Systems (OSS 2008), Springer Verlag.

[3] Postnote on Open Source Software, Parliamentary office of Science and Technology, No.242, June 2005 source: http://www.parliament.uk/parilamentary_office/post/pubs2005.cfm. retrieved on 15.4/2011.

[4] Sommerville, I., Software Engineering, 7th edition, Addison Wesley, New York 2004.

[5] ANSI/IEEE, Standard Glossary of Software Engineering Terminology, STD-729-199, ANSI/IEEE, 1991.

[6] Mall Rajib (2009), Fundamentals of Software Engineering, PHI.

[7] Rosenberg Dr. Linda, Hammer Tad, Shaw Jak, (1999), Software Metrics and Reliability.

[8] Rinette Roets, Minnaar Marylou, and Kerry Wright,(2007)Open source: Towards Successful Systems Development Projects in Developing Countries, Proceedings of the9th International Conference on Social implications of computers in developing countries, Sao Paulo, Brazil, May 2007.

[9] Feller J. and Fitzgerald B. (2000), A framework analysis of the open source software development paradigm. In Proceedings of the twentyfirst intern ational conference on Information systems', International Conference on Information Systems, pages 58–69, Brisbane, Queens-land, Australia.

[10] Johnson Kim (2001), A descriptive process model for Open source software development, University of Calgary.

[11] Raymond Eric S. (1999), The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, O'Reilly & Associates.

[12] Mockus Audris, Fielding Roy T. and Herbsleb James (2000), A Case Study of Open Source Software Development: The Apache Server. ACM

[13] Sebastian Prehn (2007) Open Source Software Development Process, Term Paper July 29, 2007, TU Kaiserslautern AG Software Engineering Seminar

[14] Robbins Jason E. Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools.

[15] Wheeler David A. (2007) Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers! Available at http://www.osepa.eu/site_pages/News/43/WhyOSS_Look_at_the_numbers_Wheeler_2007.pdf. retrieved on 15.4/2011.

[16] Marian-Pompiliu (2009), Open Source Software Reliability. Features and Tendence, Open Source Scientific Journal Vol. 1, No. 1.

[17] Stallman R. M, (2003) The GNU Manifesto. The Free Software Foundation, available at: http://www.fsf.org/gnu/manifesto.html. retrieved on 15.4/2011.

[18] Pfaff Ben, David Ken, Why open source software is better for society than proprietary closed source software. Available at http://benpfaff.org/writings/anp/oss-is-better.html retrieved on 15.4/2011.