ISSN No. 0976-5697

Volume 3, No. 1, Jan-Feb 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Some Observations on Bug Fixing Process and Defect Density of Open Source Software

Vinay Tiwari*, Dr. R.K. Pandey
University Institute of Computer Science and Applications
R.D. University, Jabalpur, India
vinaytiwari999@gmail.com

Abstract: With the success of open source software, software development is now categorized by two development methodologies. First is the traditional closed source or proprietary software development where software development is carried out in a discipline environment and in systematic manner and the other one is open source development where no definite development methodology exists. Opponents of open source software argued that open source software is not developed by following the software engineering principles like project planning, analysis, system level designing, testing etc., causes more defects compare to proprietary software. Similarly it is also argued that, since there is no post release appropriate technical/customer support as in the case of proprietary software bug fixing is poor in the open source software. On contrary OSS proponent believes that due to active involvement of internet user in online forums of OSS projects, identification and fixing of bugs is much faster. This paper analyzes the arguments made by both the communities and compares the defect density of open source software vs. proprietary software on the basis of available data and also analyzes the bug reporting and fixing process of open source software. Results of various surveys and analysis results on bug count, defect density of OSS by various researchers/agencies are also incorporated in defect density analysis. Various views of researchers on the bug fixing process of OSS are studied and analyzed and a theoretical study is made to examine the defect density and bug fixing process of Open Source Software.

Keywords: Open Source Software (OSS), Proprietary Software, Bug Fixing, Bug Repository, Bug Tracking, Software Defect Density

I. INTRODUCTION

The open source software during the last decade has got phenomenal success among the software users and developers. The story starts with the success of Linux, Apache and MySQL and nowadays, thousands of open-source software packages can be found and freely downloaded online. Open Source Project Hosting Websites like SourceForge, Google Code, GitHub, Codeplex, Launchpad etc. not only providing open source packages to the users but also providing development platform to the developers and still thousands of open source projects are in developing stages at these sites. The successful development of open source software is because of the growth of the Internet, which makes possible the collaboration among programmers on a much larger scale that was possible before. 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, [01] whereas in the conventional commercial software the end product is in the form of binary object code and the source code was assumed to be private information.

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 [02]. Software development process is now split into two development models. First is the conventional or closed source software development model, where the software is developed in controlled environment and by following strict software engineering principles. Here source code is owed by the company or individual and only binary codes are distributed under a licensing agreement to authorized users. These software are also referred as proprietary software. Second is the open source software development model where the software is not necessarily developed under controlled environment or by following traditional software engineering processes. Here source code is also released with the binary code and Users and developers have a license to share, view, use and modify the code and to distribute any improvement they make. However, open design is not an idea that everyone accepts, even now. Opponents of open source software 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' [3]. Whereas people in the open source and free software community often argue that making source code available to all is good for security. Users and experts can pore over the code and find vulnerabilities [4].

Development methodology of open source software is also quite distinct from that of traditional software development methods. For example software design before the development and software testing before the release is hardly carried out in the OSS development. Which leads the release of buggy software than the proprietary software as most of the bugs can be swallowed during testing phase. This paper theoretically analyzed the arguments made by both communities in the published reports, and trying to find out the answers to questions such as: how the defects/bugs reported and removed in Open source systems, whether the open source software has more defects as compare to their counterpart, defect densities of open source and proprietary software, reasons for less or

more bugs on OSS as compared to proprietary software. The comparisons are made on the basis of available data released by various researchers/research agencies and various vies of researchers are studied and analyzed in support of various arguments.

II. OPEN SOURCE SOFTWARE DEVELOPMENT

As per open source initiative [01] "open source is a collaborative development method for software that harnesses the power of distributed peer review and transparency of process to develop code that is freely accessible. The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in". Von Hippel [05] clarified that "Open source software development is a unique form of innovation. The developers—especially users—engage in innovation, development and consumption of a product without the direct involvement of manufacturers". Proprietary model of traditional software companies provide only binary code to the users and withhold source code, in contrast to this, open-source software is distributed under nonrestrictive licensing terms that generally include access to the source code, so that the users can study, modify, or improve it and further share the knowledge among individuals and group of people as well. Open Source software development is a different, somewhat orthogonal approach to the development of software systems where much of the development activity is openly visible, development artifacts are publicly available over the Web, and generally there is no formal project management regime, budget or schedule [06].

The Cathedral and the Bazaar [07] 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 [08] the usually mentioned main differences between commercial and open-source projects are:

a. Open-source systems are built by potentially large numbers (i.e., hundreds or even thousands) of volunteers.
b. Work is not assigned; people undertake the work they choose.
c. There is no explicit system-level design, or even detailed design
d. 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. The main strength of open source development is a well-defined community with common interests which is involved either in continuously evolving its related products or in using its results [09]. OSS is developed by loosely organized communities of participants located around the world and working over the Internet and remarkably, most participants contribute without being employed, paid, or recruited by the organization [10]. Open Source developers have typically been end users of the open source software they develop and sometimes many end-users often participate in and contribute to OSS development efforts by providing feedback, bug reports, and usability concerns. Thus Open source development is oriented towards the joint development of a community of developers and users concomitant with the software system of interest. OSS development is often less structured but the users of the systems are encouraged to directly participate as part of the development community.

In OSS development, every user has access to the source code and can thus directly participate in the continuous improvement of the software package. It is not necessary that every user be developer they can participate in OSS development by various roles depending upon their skills, technical expertise and level of involvement. Users can involved by providing feedback, helping new users, recommending the project to others, testing and reporting or fixing bugs, requesting new features, writing and updating software, creating artwork, writing or updating documentation, translating etc. All of these contributions help to keep a project active and strengthen the community. The project team and the broader community will therefore welcome and encourage participation, and attempt to make it as easy as possible for people to get involved. Depending upon the responsibilities OSS development community is classified as [11] Core developers - writing most of the code and generally responsible for software architecture, co-developers contributing code infrequently or only on some part of the project, active users providing feedback and bug reports as well as participating in discussions and helping each other in using the software, Passive users who just use the program. The success of the open source project attributed to the large spheres of co-developer and active users who find and solve various issues in the software.

III. WHAT IS BUG/DEFECT

"Bug" the computer software definition, "an unexpected defect, fault, flaw, or imperfection." A software bug is the common term used to describe an error, flaw, mistake, failure, or fault in a computer program or system that produces an incorrect or unexpected result, or causes it to behave in unintended ways. The term "defect" refers to something that is wrong with a program. A defect, in fact, is anything that detracts from the program's ability to completely and effectively meet the users needs. A defect is thus an objective thing. It is something one can identify, describe and count. According to Webster [12], defects and bugs are "sort of the same thing". The common software practice of referring to software defects by the term "bugs". Presence of hidden bugs or program code defects is a major problem with software.

Studies have shown that it is virtually impossible to eliminate all bugs from large programs. Bugs in software are inevitable irrespective of software development methodology. These bugs are caused by several reasons like complex code, human error i.e. mistakes or error made in program source code, poor design logic, lack of testing, Vague or incomplete requirements or Misunderstanding of the requirements, Misapplication of technology, Compromises made in design to meet delivery schedules etc. Few are also caused by compilers producing incorrect code. It is noticeable that bugs may or may not causes failures. Some bugs have only a subtle effect on the program's functionality [13], and may thus lie undetected for a long time. For example, defects in dead code will never result in failures. More serious bugs may cause the program to crash or freeze leading to a denial of service. Others qualify as security bugs and might for example enable a malicious user to bypass access controls in order to obtain unauthorized privileges. According to the pentagon and the software engineering Institute at Carnegie Mellon University, there are typically 5 to 15 bugs in very 1000 lines of code while commercial software typically has 20 to 30 bugs for every 1,000 lines of code. A measuring term Defect Density is used to compare the relative number of defects in various software components. Defect Density [14] is "amount of defects found in total length or size of program code". It is measured by number of confirmed defects detected in software/component during a defined period of development/operation divided by the length or size of the software/component. Defect density is generally compared by number of defects per thousand lines of source codes (KLOC). For example The NASA has a defect density of 0.004 bugs/KLOC. Measuring defect density is the easiest way to judge whether a program is ready to release.

IV. BUG MANAGEMENT IN OSS

Software defects are inevitable and it is a common practice for software to be released with unknown / known bugs. There are various reasons for not fixing of bugs in the released product like, lack of time to developers, fixing bug could be expensive or delay finishing the project or fixing may bring the chance of introducing new unknown bugs or developers think the bug is non critical and may be fixed in the subsequent release/patch.

A suitable bug management process can minimize the number of defects in software. Bugs management is the process of reporting and tracking the progress of bugs/defects from discovery through to resolution. Bug management process focus on the preventing the defects, catching the defects as early as possible, and minimizing their impact on the project. Most big software projects maintain two lists of "known bugs"— those known to the software team, and those to be told to users. Unknown bugs are reported by the customer when having problem with the released product. User send bug report through customer support to its developers, who, they hope, eventually provides some kind of solution i.e. a bug fix. Since all commercial proprietary software companies distribute their software products in compiled form i.e. binary code, customer has to depend upon software manufacturer monopoly on bug fixing. But often within proprietary software companies bug fixing is not given a high priority unless it creates a significant commercial issue. Bugs do not get fixed unless this brings a profit, and upgrades become expensive [15]. Often the people experiencing a bug are a minority of the users and the users are so locked in to using the software that they have to wait until the software company decides to fix the problem, nobody else can see the bug in the code so nobody else can fix it. Open source software is released with the source code and user have a choice to control the upgrade process and to decide which bug to fix and when. In the open source world there is often large communities around popular projects and many people who may not have the time to develop software full time, can devote some time to finding and indeed fixing bugs. In the open source world bug finding and fixing is near enough a sport enjoyed by many.

In earlier open source projects, bug reporting and tracking was done through the emails, IRC channels, or through instant messaging. Interested contributors and wide spread communities of the project discuss about patch reviews, design decision, project planning, future plan etc. All source changes are submitted as patch files to the developers' mailing list, when those interested manually apply them and test on their own systems. If the core developers approve it, the patch is then eventually committed to the source repository or incorporated into the next version placed on an ftp site [16]. Today open source projects uses more sophisticated bug tracking system or web-based application bug repositories that keep track of the change request and bugs found in a system. These bug repositories are used to report and track the problems of the software system, keep track of the change request, bugs found in a system and the potential enhancements. Proponents of open source software development believe that allowing the users of the software to easily report, and sometimes help fix, bugs improves the quality of the software produced [17]. In these bug repositories the user of the software has full access. Most Big open source project have associated with their own bug repository while other may have open bug repository like Bugzilla, Mantis, Track Jira etc. There are several potential advantages to the use of an open bug repository, these repositories not only providing bug modification progress to the developers but also ease of reporting bug to the users. This helps in identifying the more problems with the system, more problems might be fixed because more developers might engage in problem solving, and developers and users can engage in focused conversations about the bugs, allowing users input into the direction of the system [18].

A bug report in a open source projects often contains the request ID, title of the bug report, the description of the bug which contains the effects of the bug and the necessary information for a developer to reproduce the bug, possible fixes or when another bug report is marked as a duplicate of this report. Figure 1 shows the portion of a bug report at sourceforge.net. Reporters and developers sometimes may also provide attachments to reports. These attachments provide additional information about the bug, such as a screenshot of the erroneous behavior. This step is sometimes

known as bug gathering. During bug gathering reported bugs may be duplicate, provide incomplete information or may not represent real defects. [19]. Therefore in the second step such noises are removed. This step is known as bug filtering. Final phase is the bug analysis where the filtered data is organized into bug-frequencies for fixed time periods.

Request ID	Summary	Open date	Priority	Status	Assigned to
1570891	can't set a playback position in FLV videos	2006-10-04 18:43	5	Open	Nobody
1565875	mencoder mp3lame option	2006-09-26 17:39	5	Open	Nobody
1565868	mencoder -oac copy	2006-09-26 17:28	5	Open	Nobody
1564167	Use second monitor bug	2006-09-23 18:16	5	Open	Nobody
1531952	screen resizing	2006-07-31 19:37	5	Open	Nobody

Figure 1: A Portion of the bug report at sourceforge.net source [19]

Bug priority refers to the need as to how urgently bug is required to be fixed. It describes the importance of the bug. There are five levels of Bug Priority, Level 5 Immediate: Bug should be fixed as early as possible as it is blocking development/testing work. Level 4 Urgent: this type of bug; blocks the usability of the large portion of the product and must be fixed before the next planned release. Level 3 High: Seriously broken, but not as high impact should be fixed before next major release. Level 2 Normal: Either a fairly straightforward workaround exists or the functionality is not very important and/or not frequently used. Level 1 Low: The bug is not all that important. Bug priority may change according to the schedule of testing.

When a bug report is submitted its status is set to either NEW or UNCONFIRMED, depending on the conventions of the project. Once a developer has been either assigned to or accepted responsibility for the report, the status is set to ASSIGNED. Assigned Bug is resolved by resolutions Invalid: if the bug is in some way not valid, Duplicate: if the bug is repeated more than once, Fixed: The bug is checked and tested, Wontfix: The bug is described a bug which will never be fixed, Worksforme: all attempts at reproducing this bug were futile, and reading the code produces no clues as to why the described behavior would occur. When bug is duly fixed its status is set to be verified. If the bug is detected again its status is reopened and bug status is set closed when the bug is fixed and confirmed its absence.

Akinori Ihara et el [20] has explained the bug modification process using a bug tracking system is represented in figure 2. Bug modification process consisting of three different phases. First one is untreated phase which is focuses on a sub-process where bugs are reported into a bug tracking system but have not been accepted nor assigned to anyone. The modification phase is a second phase which is a sub-process where bugs are substantially modified. In this phase, a reported bug is accepted to be fixed and then assigned to developers. If the developers finish to modify the bug, the state of the bug transits to "bug resolved". The final phase is the verification phase, which is a sub-process where members in charge of quality assurance verify that modified bugs are correctly resolved.

V. OBSERVATIONS ON BUG AND DEFECT DENSITY OF OSS

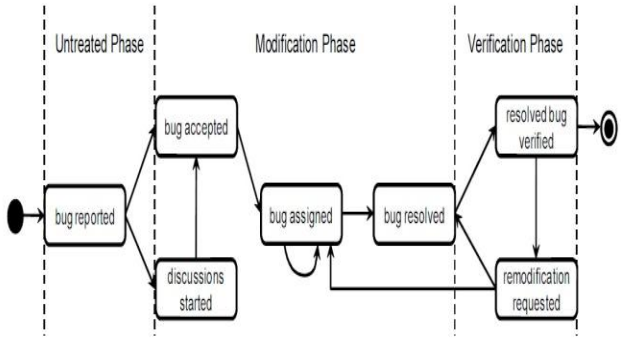

Figure 2: Bug modification process using a bug tracking system source [20]

The claim of open source community that open-source development results in better and more secure software was first examined by the Code-analysis firm Coverity [21] founded by Stanford university Computer Science Research Center. Five Stanford University computer science researchers after four year analysis of the 5.7 million lines of Linux source code they found that the Linux kernel programming code is better and more secure than the programming code of most proprietary software and had far fewer defects than the industry average. Their approach reported 985 defects in the 5.7 million lines of code in the, that make up the Linux kernel, well below the industry average for commercial enterprise software. According to data from Carnegie Mellon University, a typical program of similar size would usually have more than 5,000 defects. Windows XP, by comparison, contains about 40 million lines of code, with new bugs found on a frequent basis. The study identified 0.17 bugs per 1,000 lines of code in the Linux kernel. Of the 985 bugs identified, 627 were in critical parts of the kernel. Another 569 could cause a system crash, 100 were security holes, and 33 of the bugs could result in less-than-optimal system performance. Another newer report of Coverity in 2005, found that defect densities were very low and had even gone down further. Their follow-up analysis of Linux kernel 2.6.12 found that all six critical defects they had found in their earlier study of Linux kernel 2.6.9 had been fixed. This study found an average of 0.16 defects/KSLOC, down from 0.17 defects/KSLOC, even though the amount of code had increased, and "Although contributors introduced new defects, these were primarily in non-critical device drivers."

On Feb 11, 2003 Reasoning (A software research firm formed at Stanford University) published a study comparing the Linux TCP/IP stack to commercially developed TCP/IP stacks [22]. This comparison showed that an active, mature Open Source project may have fewer defects than similar

commercial projects. Specifically, Reasoning lined up the Linux TCP/IP implementation from the 2.4.19 Linux kernel against five commercial implementations. The company used automated tools to look five kinds of defects in code: Memory leaks, null pointer dereferences, bad de-allocations, out of bounds array access and un-initialized variables. Reasoning found 8 defects in 81,852 lines of Linux kernel source lines of code (SLOC), resulting in a defect density rate of 0.1 defects per KSLOC. In contrast, the three proprietary general-purpose operating systems (two of them versions of UNIX) had between 0.6 and 0.7 defects/KSLOC; thus the Linux kernel had a smaller defect rate than all the competing general-purpose operating systems examined. The rates of the two embedded operating systems were 0.1 and 0.3 defects/KSLOC, thus, the Linux kernel had an defect rate better than one embedded operating system, and equivalent to another.

In 2003, Reasoning, Inc also performed a defect analysis of the Apache web server and Tomcat, which is a mechanism for extending Apache with Java Servlets, by using their defect discovery tool. For Apache, the tool found 31 defects in 58,944 source lines, a defect density of 0.53 defects per thousand lines of source code (KSLC). In a sampling of 200 projects totaling 35 million lines of code, 33% had a defect density below 0.36 defects/KSLC, 33% had a defect density between 0.36 and 0.71 defects/KSLC, and the remaining 33% had a defect density above 0.71 defects/KSLC. For Tomcat, the tool found 17 software defects in 70,988 lines of Tomcat source code. The defect density of the Tomcat code inspected was 0.24 defects/KSLC.

In a similar manner to the previous studies, on December 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 tools found a defect rate of 0.57 defects/KSLOC, over six times the error rate.

Mockus et el and later Dinah-Trong et al[23] measure and compares the defect density of apache code and FreeBSD with four commercial projects and find that the defect density of Apache and FreeBSD is smaller than the commercial systems after the feature test.

In an another report by Coverity after analyzing the software quality of popular open source project, The average defect density for these 32 open source packages that analyzed by them was 0.434 defects per thousand lines of code. While most popular open source packages Linux, Apache, MySQL, and Perl/PHP/Python showed significantly better software security and quality above the baseline with 0.290 defects per thousand lines of code. Coverity's recent open source integrity report published in 2010 on smartphone based popular operating system Android Foryo, says that The Android kernel has better than industry average defect density (one defect for every 1,000 lines of code). Android Kernel 2.6.32 has about half the defects that would be expected for similar software of the same size.

Ioannis Samoladas et el [24] studied almost 6 million lines of code, tracking several programs over time, using the maintainability index (chosen by the Software Engineering Institute as the most suitable tool for measuring the maintainability of systems). Using their measurements, they concluded that OSS "code quality appears to be at least equal and sometimes better than the quality of [closed source software] code implementing the same functionality." They conjectured that this "may be due to the motivation of skilled OSS programmers...".

VI. DISCUSSION

Any software tends to be buggy because of several causes. Apart from other some of the causes from which open source gain the advantages are as follows:

a. Miscommunication: Lack of communication or miscommunication between parties at any stage in the development phase is the common reason for the software defects. In proprietary development environment developers develop system for other. In the requirement gathering phase communication error like vague, incomplete, ambiguous or non-specific requirements causes defects in the software, because the developers would have to deal with the problems that are not clearly understood. This situation is not arises in the open source development as the developers are developing software in which they have interest and clearly understand the problem and knows what is to be develop.

b. Last minutes changes: Last minute change in the requirement, change in tools/ platform, late design changes can require last minute code changes, which are likely to introduce errors. In OSS projects generally development starts with the need of developers so the problem is definite and well understood at the beginning of the project by the developers. Therefore these types of last minutes changes are not occur in OSS development causing more reliably ands less buggy code generation.

c. Commercial pressure: OSS 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.

d. Scheduling or Time pressure: The fixed schedule imposed on the software developers also degrades the quality of the software [25]. In a proprietary development Customers, business owners or managers want things done fast and often have little awareness of how long it may take to create a piece of software, giving the development team a highly unrealistic deadline to complete a project. Because of the time pressure it is probable that compromises are made in requirement/design to meet delivery schedules. Programmers are not having enough time to design, develop and test their code and forcing them to release software that's possibly not ready which leads to errors and defects. On the other hand open source projects are

largely immune from "time-to-market" pressures [26] Programmers/developers have no deadlines or scheduling pressure. It is released if and when it is ready, which generally means that there has been an honest attempt to identify unknown bugs and that all known bugs have been fixed. A system is released only when the project owners are satisfied that the system is mature and stable.

e. Programming Error or Poor coding practices: In a proprietary development programmers can make mistakes because of their poor training, lack of interest in the project, pressure to quickly turning out the code resulting unclear and non understandable code. It's tough to maintain and modify code that is badly written or poorly documented; the result is bugs. Whereas in Open source development developers join projects because of their interest, and write code with more care and creativity because developers are working only on things for which they have a real passion. The absence of code development pressure results a well written and documented code.

f. Human Factor: Bugs may also be introduced due to distractions, high stress level, and low knowledge level of technology. OSS development is a part time activity and developers working only on the technologies in which they have confidence and participated in open source projects for challenge, reputation building, improving skills, altruism and for fun.

In the open source model often it is argued that source code availability allows faster software evolution. The idea is that multiple contributors can be writing, testing, or debugging the product in parallel, which supposed to accelerate software evolution. Open Source proponent often quote the "Linus Law" given by Eric Raymond "Given enough eyeballs, all bugs are shallow" means the more people who can see and test a set of code, the more likely flaws will be caught and fixed quickly. This maxim can be verified by the series of study and analysis on open source and commercial software comparisons, conducted by reasoning. Reasoning found that defect density of open source software is comparable to the proprietary software. In the first study the Linux TCP/IP stack was compared to commercial developed TCP/IP stacks and the comparison showed that an active, mature open source project may have fewer defect than a similar commercial project. In the second study, where pre release Apache http server v2.1 was compared with the less mature commercial code and it was found that open source and commercial software start at a very similar defect density. i.e. open source and commercial software have a very similar quality. In the third open source study mature Tomcat 4.1.24 application server code was compared with proprietary code it was found that Tomcat showed a defect density similar to proprietary code at a similar point in the development life cycle. In the next open source code inspection project on the request of developers, MySQL4.0.16 open source database was compared and it was found that the defect density of MySQL is about 6 times lower the average of comparable proprietary projects

Reasoning [27] validated these findings by plotting a graph between defect density and time to observe the approximation of the change in defect density over time. Given the limited data, Reasoning sees Open Source as being faster, on average, than commercial efforts at removing defects from software. This is not as expected, since commercial software companies often invest considerably in testing tools and time in order to meet reliability requirements demanded by their customers. It should be noted that Open Source can end up with fewer defects. Because the new evidence shows that both development environments are likely to start with a similar number of defects, the core of the difference must be after development starts. In that time period, the main difference is the number of developers actually looking at the code. Commercial software companies are more likely to have sophisticated tools; however, they are unlikely to have achieved the same level of peer review that Open Source code can achieve, which is critical in finding defects.

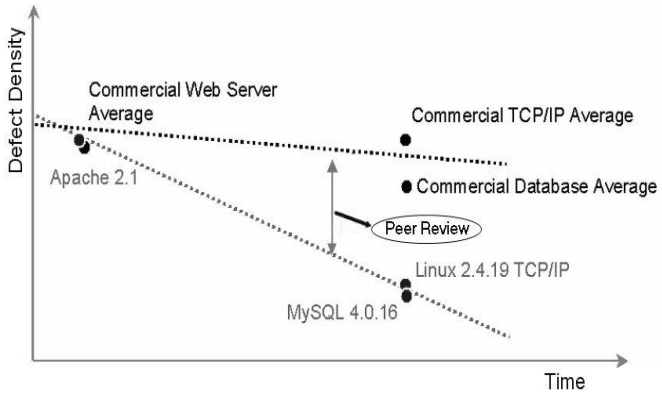

Figure 3: Benefits of peer review source [27]

Thus in open source development peer review process drive excellence in design since large amount of developers globally contributing and analyzing the code, making OSS secure and constantly increasing the quality as open source software code is available publicly. In the case of Linux, about 1,200 programmers have contributed bug fixes and other code. This means if a bug is reported in Linux, a couple dozen programmers begin looking for it, and many bugs are corrected within hours. The quality and security of Linux is accelerate through the addition of several "automated" eyes. This Linux axiom points to the fact that when a bug becomes an issue, many people have the source code, and it can be quickly resolved without the help of a vendor [28]. Various studies on successful open source project reveals that "In successful open source developments, a group larger by an order of magnitude than the core will repair defects, and a yet larger group (by another order of magnitude) will report problems".

Openness in the OSS is another key benefit. Openness of the source code allows faster software evolution since multiple contributors writing, testing, or debugging the product in parallel and sharing development efforts among the group members. Openness of the communities allows users who experience a bug in the software to locate that bug and to fix it accordingly. The openness of the review process usually makes it possible to resolve the conflicts through discussion.

Community members can work together based on the recorded information to resolve conflicts. Opponents of OSS software 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" [29]. But the fact is looking at the code , which can often be more than 1,00,000 lines is a very inefficient way of finding insecurities, indeed a lot of people have found many insecurities in Internet Explorer and Windows without seeing a line of code. Anderson proved a theorem that it does not matter whether the system is open or closed. Opening a system enables the attacker to discover vulnerabilities more quickly, but it helps the defenders exactly as much.

Proprietary software often passes through a beta testing before the final release. But this beta testing is not effective at correcting remaining bugs because the team that developed the software is the same one attempting to fix bugs and the remaining bugs can not be easily found as the developers has too many built in assumptions about the code. Developers in proprietary organization are mainly not users and therefore they do not know which functionalities to develop or improve first or simply where the bugs are. On the contrary, open-source communities benefit considerably from a "users as innovators" organization [30], and attract numerous heterogeneous developers which, using their own idiosyncratic experience, correct various bugs and suggest various new developments. In OSS development active user community which is involved in bug-fixing, testing, bug-reporting, documentation, release management etc., helps the developers community to stay concentrated on their work and also enables extensive testing and efficient bug fixing. Participants of project are generally well aware of who the experts are for a particular type of bug. Bug fixing is allotted to particular expert group and severe defects may also be fixed within hours of their detection. Sometime micro-competition may occur when multiple participants work on a single bug which also resulting a quicker bug fixing. Raymond also suggests that debugging is even more efficient when users are co-developers, as is most often the case in open-source projects. In OSS development most developers start out as users and therefore guide their development efforts from the user's perspective.

According to CIO magazine's report. [31] The average time to resolve an application problem is 6.9 days for enterprise developers and 6.7 days for software vendors. Ten percent of those problems take 10 days to solve. Evans Data Corporation (EDC) after conducting a survey of several hundred open-source and Linux developers reported that "The average time between discovery and solution of a serious bug, for 36 percent of open-source developers is less than eight hours. Hours, Not days, Not a week". Thus the open-source development process is much, much faster at fixing bugs than the proprietary-software development process.

VII. CONCLUSIONS

In this paper studies are made on the defect density of OSS as compare to proprietary software and bug management practices in open source software. Open-source software development presents an approach that challenges the traditional, closed-source development approaches. For the some of the popular open source software, various analysis and studies suggest that the open source software having defect density below the industrial average. Although this may not prove that OSS will always be less buggy or of highest quality, but it clearly shows that OSS can be of high quality. Although, open source projects not following the traditional development methods, do not having well-designed development process, planed resources but open source communities are constantly creating and improving their working methodologies.

The peer review process drive excellence in design and open source development model benefits from the "many eyes" approach. Bug identification and reporting is much better and bug fixing is fast, as compare to proprietary software. Only open source has a plan for fixing bugs in the environment where they're discovered. Bug repositories also playing important role in OSS development. Information stored here is useful for the developers and researchers to understand the development process like development methodology, roles of people, level of involvement, bug reporting, automated bug assignment etc. Studies also suggest that in later versions, OSS having less or even zero bugs because of quick identification fast fixing of bugs. This causes the popularity of OSS among software users. In alone SourceForge site software like VLC media player, 7-Zip, eMule, Filezilla, Smart package of Microsoft's core fonts, Portable Software/USB having more than 10,00,000 weekly downloads. Not only users, developers also associating with OSS movement to enhance their skills. As of July 2011, the SourceForge repository hosts more than 300,000 projects and has more than 2 million registered users. In last 2-3 years open source project involvement has increased more than 50%.

Software development companies are also taking part in the open source movement and using OSS component for their product. They usually use the beta versions of OSS; report bugs or fix bugs and often release software with added functionality. Android operating system for smart phone is a good example of this. The whole operating system is open source, but OEMs can add proprietary software on top of it for custom applications for their devices. Analyst firm Gartner also estimated that by next 2 years, at least 80% of commercial software packages would include elements of open source technology. Clearly the future of the software is the hybrid form of OSS and proprietary software.

VIII. REFERENCES

[1] Open source Initiatives, http://www.opensource.org/docs/osd.

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

[3] K Brown, \Opening the Open Source Debate", 2002, Alexis de Toqueville Institution, at http://www.adti.net/html_files/defense/opensource_whitepaper.pdf.

[4] Ross Anderson, Open and Closed Systems Are Equivalent (That Is, in an Ideal World)., Perspectives on Free and Open Source Software, p127,2005.

[5] Von Hippel, Eric (2001), "Innovation by User Communities: Learning from Open Source Software." MIT Sloan Management Review, Summer 2001.

[6] Yi Wang, Defeng Guo EMOS/1: An Evolution Metrics Model for Open Source Software, source internet.

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

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

[9] Gacek, C. and Arief, B., The many meanings of open source, IEEE Software, Vol. 21, No. 1, pp. 34-40, 2004.

[10] C horng-Guang W., James H. G., Clifford E. Y, An empirical analysis of open source software developers' motivations and continuance intentions, Information & Management, Vol. 44, No. 3,pp. 253-262, 2007**.**

[11] Otso Kivekas, Free/Open Source Software Development: Results and Research Methods,Master thesis, pp. 18,2008.

[12] http://www.websoftwareqa.com/2011/01/defects-vs-bugs-are-they-different/, accessed on nov. 2011.

[13] Software bug - Wikipedia, the free encyclopedia ,en.wikipedia.org/wiki/Software_bug .

[14] Linda Westfall, Defect Density, http://www.westfallteam.com/Papers/defect_density.pdf.

[15] Yogesh Suman & A K Bhardwaj, Open Source Software and Growth of Linux: The Indian Perspective , DESIDOC Bulletin of Information Technology , Vol. 23, No.6 , pp. 9-16, November 2003.

[16] D. Cubranic. "Open-source software development." Workshop on Software Engineering over the Internet, held as part of the IEEE/ACM International Conference on Software Engineering (ICSE'99), 1999.

[17] A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of open source software development:Apache and mozilla. ACM Trans. Softw. Eng. Methodol., 11(3):309–346, 2002.

[18] John Anvik, Lyndon Hiew and Gail C. Murphy, Coping with an Open Bug Repository, downloaded from citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.116.

[19] Cobra Rahmani, Harvey Siy, Azad Azadmanesh, An Experimental Analysis of Open Source Software Reliability, www.cse.buffalo.edu/srds2009/.../cobra-open-source-srds09-camera.

[20] Akinori Ihara, Masao Ohira, Ken–ichi Matsumoto, An Analysis Method for Improving a Bug Modification Process in Open Source Software Development, IWPSE-Evol'09, August 24–25, 2009, Amsterdam, The Netherlands.

[21] http://www.coverity.com/html/research-library.html #CaseStudies, accessed october 2011.

[22] http://www.reasoning.com/downloads.html, accessed october 2011.

[23] Trung Dinh-Trong and James M. Bieman, Open Source Software Development: A Case Study of FreeBSD Proc. Tenth Int Software Metrics Symposium 2004

[24] 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/ Why OSS_Look_at_the_numbers_Wheeler_2007.pdf. retrieved on 15.4.2011.

[25] 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.

[26] M. W. Godfrey and Q. Tu, "Evolution in Open Source Software: A Case Study," Proceedings of International Conference on Software Maintenance (ICSM'00), 2000.

[27] http://www.reasoning.com/downloads.htmlMySQL_White_Paper4 retrieved october 2011.

[28] Crowston, K. & B. Scozzi. 2004. Coordination practice within FLOSS development teams: the bug fixing process. Paper presented at the first International Workshop on Computer Supported Activity Coordination, Porto, Portugal.

[29] K Brown, \Opening the Open Source Debate", 2002, Alexis de Toqueville Institution, at http://www.adti.net/html_files/defense/opensource_whitepaper.pdf.

[30] Von Hippel E. (1988), The sources of innovations, MIT Press.

[31] Matt Asay, Open-source vs. proprietary software bugs: Which get squashed fastest?, accessed from: http://news.cnet.com/8301-13505_3-9786034-16.html on october 2011.