CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

The major objective of this chapter is to provide background information which is required for understanding the open source software development and the impact of release management towards quality improvement in open source software project management. This chapter presents a short description of the various studies carried out by researchers related to the significance of OSS, the method of OSS development, release management approach, key factors of OSSPM and the quality standards related to OSS.

Sanders (1998) in his study on ‘Linux, Open Source and Software’s Future’ observed that software companies must find a way to meet often conflicting requirements of developing software faster while making it more robust. Linux appears to be the most preferred open source software by programmers. One solution may be to abandon the corporate hierarchies, project-specific development teams, and milestone-driven Gantt charts - to embrace anarchy. Linux’s evolution defines a development model that creates a faster, leaner, and more reliable product than mainstream competitors, and does so essentially for free. The source code is open; and the four most critical pieces of infrastructure that make the internet work include - Bind, Perl, send mail and Apache - every one of these is open source, and super reliable.
Open source software development has its own strengths as well as its weaknesses. The legions of technically adopt developers and programmers who freely contribute to open source software are profoundly more skilled than the average computer user. Indeed, many developers share the simple interfaces that make mainstream programs related to the non-technical development approach.

Sanders also opines that open source software projects benefit directly from the products they help develop, which gives them still greater incentive to contribute. Open source developers enjoy peer recognition, intriguing problems, and products better suited to their needs because they’ve helped to develop them.

These advantages account for much of the effort devoted to internet components and to the Linux OS, and could easily motivate work on open source utilities and development tools. But, it strains the open source paradigm to think volunteer developers will create commercial-grade applications. He concludes that in mainline software development, open source promises a vast pool of elite programming talent if one can integrate the free-form of resource into a more structured development methods.

Yamauchi et al (2000) in their study entitled that ‘How Open-Source Software Succeeds’ highlighted the prevailing competition and found that OSS developers rarely meet each other as they depend mainly on electronic media. A practice of open-source software development is perceived to be significant to computer supported cooperative work research for mainly two reasons. First, software development is traditionally a coordination-intensive process and has attracted several OSS researchers; and secondly, open-source software is developed by geographically dispersed programmers with the aid of collaborative technologies such as electronic mails. The study used ‘observation method’, supplemented by ‘interview
techniques’ for quantitative analysis of data. The central focus was on the use of electronic mode of communication technologies.

Based on observation and interviews, quantitative analyses of two open-source projects were done to address the problems of dispersed collaboration of volunteers. The findings of the study show ‘how open-source development avoids limitations of dispersed collaborations and address the sources of innovation in open-source’. The findings also suggest that spontaneous work coordinated through electronic mode of communication is effective and rational.

Dempsey et al (2002) in their study entitled ‘Who is an open source developer’, found that software source code as a form of scientific knowledge, and just as scientists publish so that other scientists can build on their results, computer scientists must publish their source code in order to foster continued innovation in computing for the welfare of the society through utilizing the software on free of cost. The researchers provide a quantitative profile of a Linux community of open source software developers, individuals submitting non-juried contributions to a repository of Linux materials. Linux plays an increasingly significant role in the business plans of established computing companies, in university research labs, and in the development of new companies focused on Linux support and integration issues. With reference to the Linux Kernel OSS project leverages, the researchers observed that the power of Internet communication lies in bringing together a large number of developers in a coordinated effort.

In the conclusions of the study, the following issues were identified as a basis for developing and evolving software for the OSS community: The systems and applications categories of OSS are by far the largest areas of contributions. Linux developers who contribute to the Archives are generally serious type of individuals. The Archives continue to grow at a very fast rate,
especially since 1993, showing that there is an ongoing dedication to the project even if that dedication comes from a shifting set of contributors. Open source empowers individual programmers to participate in a large programming community in a meaningful way. They also indicate people (OSS volunteers) do their best work when they are passionately engaged in what they’re doing.

Madey et al (2002) in an empirical study on ‘The open source software development phenomenon: An analysis based on social network theory’, recognized that the OSS movement is a phenomenon that challenges many traditional theories in software engineering, business strategy, and IT management. The researchers claimed that open source development produces more bug-free code, faster than closed proprietary developed code. The volunteers of OSS team have developed software which are part of a successful virtual software development project, and not for commercial purpose. Such team members often come from different geographical regions and rarely meet one another. According to the researchers, a significant portion of internet e-commerce runs on OSS, and thus many firms have little choice but to trust a mission-critical e-commerce approach to run on such softwares including new types of socio-technical problems. They hypothesized that open source software Development can be modeled as self-organizing, collaboration, social networks.

The conclusion supports the speculation that the open source movement is not a random graph (i.e., new nodes attach to existing nodes with uniform probabilities), but a graph displaying preferential attachment of new nodes (i.e., some nodes have higher probability of attachment that others). This typically happens under the situations of positive feedback or increasing returns to volunteers.
Hertel et al (2003), carried out an exploratory study on ‘Motivation of software developers in Open Source projects’. The objective of the study was to explore the motivation of such persons who spend considerable time and effort in OSS projects. The motives of 141 contributors to a large OSS project (the Linux kernel) were analyzed with structured questionnaire. The study focused on the subsystem of team work aspects of Linux development process. This subsystem is technical and not an organizational entity. Therefore, it is not essential for the team to consist of only one OSS developer.

Measured factors were derived in relation to Linux OSS community as well as from models of social sciences. Participants’ engagement was particularly determined by their identification as a Linux developer, by pragmatic motives to improve own software, and by their tolerance of time investments. Moreover, some of the software development was accomplished by teams. Activities in these teams were particularly, determined by participants’ evaluation of the team goals as well as by their perceived indispensability and self-efficacy. It is found that the higher participants with more hours fetches more personal rewards for their Linux engagement.

Zhao and Elbaum (2003), researched the aspect of ‘Quality assurance under the open source development model’. The open source software development model has challenged traditional software development practices by generating widely accepted products (e.g., Linux, Apache, Perl) while following unconventional principles such as the distribution of free source code and massive user participation.

The researchers were of the opinion that the emergence of open source is a challenge to quality assurance approaches, claiming to be successful through techniques and principles that affect some of the current and standard software development practices.
Examining the related work on open source models, the survey methodology adopted in the study showed that OSS projects promote free distribution and provided complete access to source code. The idea is that multiple volunteer developers are involved in programming, testing and debugging the OSS product in parallel, which accelerates the software evolution.

The study explored how the software quality assurance is performed under the open source mode and how it differs from more traditional software development models. The main findings of the study include that the level of user participation in open source projects was extremely high, generating up to 20% of the changes for almost 50% of the projects, and discovering 20–40% of the faults in 20% of the projects.

Pressman (2004) discussing the issues in OSSD, describes software development as the process which is carried out before it is launched as an operational system. In traditional software development, the activities during this phase include requirements and system specification, initial design, software coding and testing. Many methodologies are available for carrying out software development. These methodologies specify critical issues like team formation, task assignment, milestone definition and cost and scheduling of the project. The failure rate of software development projects is very high. The 90% syndrome in software development projects implies that the majority of software development projects fail to meet the expected time and cost schedules (Abdel-Hamid 1988). This failure affects the overall system costs and performance of the software. Many of software development research have been focused on identifying the causes of software project failure. The community has also been long in search of a silver bullet that would help overcome this great challenge (Brooks 1995).
However, the nature of software development is such that once software code is complete and has been tested, it is ready to be operational and to be used by the computer user. It undergoes change throughout its life; and any change made to the software product after its development has been completed, it is called software maintenance (Pressman 2004; Zelkowitz et al 1979). The software maintenance includes activities like fault correction, improvement of performance or adaptation to changes in the operational environment (Pressman 2004). Software maintenance claims a large proportion of the lifecycle costs of a software system and is a large component of the Information Systems (IS).

Hence, there is a greater degree of interest in the improvement and control of the process of software maintenance; and many of the problems of software maintenance are the result of software development and design inadequacies (Schneidewind 1999; Swanson and Beath 1997).

Linus and McKelvey (2005), analyzed ‘Who is not developing open source software? Non-users, users, and developers’. In their analysis opined that the development of knowledge requires investment, which may be made in terms of financial resources or time. The OSS has challenged much of the traditional reasoning by suggesting that individuals/volunteers behave willingly and contribute to the benefit of computer users. The major share of the existing OSS system includes volunteer communities who are already active development activities of OSS community. The authors explore main issues, which include: How frequently a group of skilled people use OSS, reasons for differences among users and non-users in terms of use and attitudes; and how frequently, and why, some users contribute to OSS projects (and thereby become developers).
In doing so, the researchers consider the opportunity costs of use and development of OSS, which has been largely neglected. It was also observed that the individuals have a rather pragmatic attitude to OSS projects and that many are active in both OSS projects and community.

Robles et al (2005) in their study ‘Evolution of Volunteer Participation in Libre Software Projects’. Most of the software projects rely on the work of Volunteers; and hence, attracting people who contribute their time and technical skills is a significant factor in terms of both technical and economic premises. The involvement of volunteers gives raise to new issues that have to be taken into account for business strategies around OSS. Collaboration from volunteers is difficult to predict, but if it is ensured it may add value to a software.

The researchers observed that in the Debian operating system including a large number of applications, such as GNU tools, Mozilla and Linux Kernel, the main characteristics of the OSS distribution and the whole life of the project has been maintained by volunteers and it has grown substantially.

The investigators analyzed as ‘how many of the maintainers have remained from the beginning and what has happened to the packages from maintainer who left the project’. Results show that the growth of OSS release packages is actually bigger than that of volunteer’s contribution.

From the study, it can be inferred that the key to develop a framework lies in assessment of projects which are sustainable and considers various factors such as the number of developers, or the size of the user base. This assessment can also include simple security audits of source codes to establish whether a piece of software has sufficient quality so that the
maintenance is possible. More experienced maintainers are responsible for packages which are installed and used regularly more often.

Michlmayr (2005) in his study focused that ‘Quality Improvement in Volunteer Free Software Projects: Exploring the Impact of Release Management’ aimed to answering the basic question as ‘how volunteer projects can deliver predictable and high quality software’. Exploratory interviews with a number of free software and an open source developer were used as the tool for this study and release management has been found to be a problematic area.

Since developers mainly use development releases, they might not see the need for well tested and stable releases aimed at less technical and end users. Inadequate release management can lead to a number of problems, such as software which is out of date, breaks compatibility, or does not meet the quality standards or the requirements of users, according to the study. This study hypothesized that time-based releases offer a number of advantages to users and releases are more predictable, the development leads to more features, better code and the release schedule allows for more systematic testing of the software.

The study concluded that release management is one aspect of quality management and quality improvement in volunteer project activities. Release management is a problematic area in OSSD in which further improvements are possible.

Wasko and Faraj (2005) conducted a research on ‘why should I share: Examine social capital and knowledge contribution in electronic networks of practice’. They opined that a main feature of OSSD is its virtual development environment and essentially, CSSD projects may also sometime work in virtual development environments. However, OSSD is entirely
accomplished in the virtual environment and completely subjected to the virtual community’s norms. The practices of virtual communities are also known as computer mediated communications or electronic network of practice. It was described as “self-organizing, open activity system focused on a shared practice that exists primarily through computer-mediated communication”. The online virtual communities consist of many characteristics such as virtual, Peer-to-peer, decentralized structure and open membership and voluntary contribution; communication is an important mode for the virtual community. When comparing the differences between computer-mediated communication and face-to-face meetings, researchers found that common ground, context and trust (i.e., the factors that are highly important for an effective communication) are difficult to establish the virtual environment.

Face-to-face meetings are generally deemed as richer communications than computer mediated communication (Olson and Olson 2000). In the virtual environment, the steps and strategies of building relationship between a leader and team members are also important. Several researchers in OSSD stressed the importance of virtual community functions for the success of an OSSD project.

Computer-mediated communication is inherently impersonal and favours task-oriented and focused exchanges (Hiltz and Turoff 1993).

Thus, the OSSD project community, as a relatively stable, professional virtual community, needs to provide more motivation to its participants to share quality knowledge than other virtual communities with more volatile members.

David and Moss (2006) in their study on ‘Free and Open-Source Software: Opening and Democratizing E-Government’s Black Box’
considered the implications of the use of open-source software in government. From a constructionist perspective, the broad effects of non-proprietary software are contingent on ‘how the practice of open-source software is discursively represented and constituted’ as it is translated into new e-government systems. On these premises, an analysis of official discourse and government policy for non-proprietary software was suggested. The researchers found that the introduction of OSS projects into government system will bring more development activities. However, on the basis of system flexibility, an alternative discourse of software elements of OSSPM encompasses circumstances in which release activities are undertaken. Promoting the practice of non-proprietary software contributes to opening-up of e-governance activities and improving transparency and accountability.

Roberts et al (2006) investigated ‘Communication Networks in an open source software project’. They explored the nature of the social network and the patterns of communication that exist in an Open Source Software Development Project.

Open source software has played a fundamental role in the development of the Internet by contributing to such remarkable software as TCP/IP, BIND, Send mail, Linux, and the Apache WEB server.

The study also analyzed archival data on mail communications between developers in the Apache HTTP server project and suggested an interesting pattern of communication. They found that the core developers self-organize into three sub-groups that communicate intensely in completing the project. The analysis also reveals that a few prominent developers who are centrally located in the network are driving communications within the project.
One of the main conclusions drawn by the researchers was that the influence of communication pattern is significant on the aspects of OSS project performance are outcomes. Measures of influence and position within an OSS project’s social networks may help explicate relationships between individual developer participation and performance.

Stewart and Gosain (2006) in their article stated that, The Impact of Ideology on Effectiveness in open source software development teams was investigated by the research question as to ‘what leads to effectiveness in OSS development teams in the absence of formal controls and the importance of ideology’. The researchers formulated an approach based on the assumption that the OSS ideology motivates behaviours which enhances affective trust. These unique aspects of OSS have led many others also to question why developers contribute (Hars and Ou 2002) and how their dispersed efforts are controlled in order to result in viable outputs (Gallivan 2001). Answers to these questions may lie, partially, in the unique feature of OSS development, i.e. the ideology of the OSS development community.

Krishnamurthy (2006) investigated the domain of motivation in the context of open source software, which may be seen as fundamentally different due to the presence of unpaid programmers, implicit rather than explicit forms of control and a different methodology for software development. Since software development involves creative tasks and approaches, the motivation of open source programmers can be compared to individuals in creative software projects. The study summarizes the important trends in the research on motivation in open source and identifies variables which need to be analyzed for further research. Specifically, the current literature favours a taxonomy that considers two components of motivation – intrinsic (e.g., fun, flow, learning, community) and extrinsic (e.g., financial rewards, improving future job prospects, signaling quality). The researcher
makes a case for incorporating both elements in developing an integrative theory about OSS developers’ motivation. Three elements are identified as being unique to OSS development, namely, diversity of project structures, co-existence of companies and communities and co-existence of creative and commercial elements.

Francesc et al (2006), dealt with particulars of the OSS development process and its perceived “social” connotations and traditional ways of explaining IT adoption. Evidence show that OSS fails in many cases to displace dominant market leaders even in the case of computer user's unhappiness with the prevalent solution, while in some others OSS is adopted without a clear advantage. User communities and broader social responsibility considerations have been found to exert some degrees of influence on the IT innovators. Through the analysis of some significant cases the researchers propose a framework which helps to depict ‘under which conditions significant OSS adoption may unfold’.

Paula (2006) also conducted a micro level study on ‘IBM’s Pragmatic Embrace of Open Source’. The researcher examines IBM Corporation’s adoption of an open source software model. Though IBM began as a strong advocate of intellectual property rights for computer programs, the company has embraced open source software under the general public license in the 2000s. The company is showing its dedication to the open source movement by contributing $100 million per year to the development of Linux and other open source projects, as well as donating some of its proprietary software to strengthen Linux's ability to provide enterprise-level capabilities.

Samuelson argues that IBM’s embrace of open source may be understood in three ways: a) an anti-Microsoft strategy, b) consequence of changed business models in the software industry and c) manifestation of an
open innovation strategy for promoting faster and more sound technical advances.

Howison et al (2006) conducted a study on ‘Social dynamics of free and open source team communications’ Effective communication is a main tool for the success of OSS communities and their activities. Most members of this kind of community are geographically distributed and communicate with each other through electronic media (mailing lists, bulletin board systems or some real time communication software, for example). The purpose of this report is to simply state what the OSSD is, how people (especially users and developers) participating in open source software development community and communicating for development of software and how to improve the communication between users and developers. Statements are supported by using data collected from various articles, electronic journals, books and Internet sources.

The study concluded that a mode of electronic communication ensures coordination and rapport building between users and developers in OSS community, in order to convey message accurately and clearly between them, project coordinator is responsible to make the communication between users and developers better by applying the suitable communication skills when it’s necessary.

Golden (2007) describes that the qualitative study on ‘open source as community’ process of selecting an evaluating OSS as a time consuming exercise. An important criterion in evaluating open source software include the size and vibrancy of the community, the availability of online documentation, and access to support via mailing lists, forums, and Internet Relay Chat (IRC). One source for such information is statistics on sites such as Source Forge (http://sourceforge.net). However, the website focuses on the contribution of developers and it is important to note that contribution to an
open source community is more than just commitment to the source code base. As in proprietary software, production of good open source software also requires documentation, testing, support, training, and the incorporation of user feedback. An understanding of the activities of the volunteer community can help to answer questions such as ‘what support mechanisms are available if we release the software and how difficult will it be to install and use the software by computer users?’.

In summing up of analysis, the researcher emphasizes that in making a final release decision, it is necessary to evaluate the development and maintaining of new OSS release components, while taking into account the scope and nature of the effort involved. Businesses incorporating open source software into their products may want to consider leading the development and ongoing maintenance of a new component.

Ebert’s (2007) study on the evolution of free and open source software movement has had phenomenal impact on the industry’s evolution. This study conducted an in depth analysis of technological innovations and observed that FOSS has sparked novel business models, intellectual property strategies, and software development paradigms. The fact, that the software world we have is unimaginable without open source operating systems, databases, application servers, Web servers, frameworks, and tools is justified. While the brands of OSS such as Linux, MySQL, Apache, and Eclipse, together with their underlying software, have dramatically shaped product and service development, they facilitate competition and open markets as well as innovation to meet new challenges. In addition, FOSS also changed the way we develop software and several current engineering processes have evolved from the way open source is developed, as was perceived by Christ of Ebert.
In software technologies and processes also, open source product innovations are most visible and open source models have the potential to change the development processes. Moreover, source code accessibility facilitates independent reviews and test-driven development and code-analysis tools help in providing the quality infrastructure.

Michlmayr et al (2007) carried out a study on ‘Release management in free software projects’, emphasized that release management plays an important role in every software project. Since it is concerned with the delivery of a high quality product to end-users, coordination among distributed team of volunteers to align their work for a release activity was considered as a focal point by the researchers. OSS is also characterized by a highly iterative development model in which new development releases are typically made available very frequently.

Delays and unpredictability were considered as factors for slowness in release activities affecting the popularity and importance. The study concluded that the role of the release manager is diverse and demanding because they have to interact with a large number of different people, understand technical issues but also know how to plan and coordinate. The need for specific skills among developers has been highlighted by the study and these include community building, strong vision, communication, judgment and commitment to creating new releases. The use of version control and bug tracking systems serve as an important basis for release management as they provide a sound status for the OSS projects. Freezing, scheduling, establishing milestones, setting deadlines, building on different approaches, user testing and following a release checklist benefit the OSS projects in multiple ways.

Shaikh and Cerone (2007) conducted a study on ‘Towards a quality model for OSS and justified that software quality is more than just
conformance to a set of requirements, and represents many attributes related to each other to make a piece of software.

Software quality metrics, for instance, as measures of software quality encompasses factors such as software correctness and related attributes. Functionality, reliability, usability, maintainability, portability and efficiency, which are assessed and judged through acceptable means, are within the scope of software correctness. According to the researchers, the software correctness as a factor of quality improvement functions as a refinement technique and can be verified using formal methods, along with validation using manual and automated testing. OSS development process includes characterization of metrics and has a wider availability to an open community of peer developers, code reviewers, contributors and end-users, who are free to engage and employ a diverse range of development practices and methodologies.

Shaul Oreg and Oded Nov (2008) conducted a study on motivation which is a key factor in OSSD and examined the contextual and dispositional correlates of motivation. For the purpose of the study, a web-based survey was administered to 300 subjects of volunteers by the researchers in two prominent open source contexts: software and content in order to measure ‘how the open source project and the personal values are related to the types of motivations’. It was hypothesized that the software developers placed a greater emphasis on reputation-gaining and self-motivation. ANOVA technique was used as a methodology and it was identified that human behaviour in software development involves the idea that by drawing on the contributions of volunteers, technically superior software can be created. The study reveals the OSSD is based on a collaborative effort where software is created by a community of volunteers or members of organizations who support the open source software movement. The relationship between open
source context and motivation is established on the criteria of personal values and gaining of reputation by volunteers.

The findings indicate that by contributing to open source software initiatives volunteers maintain their status, identity and abilities. Intrinsic and extrinsic motivation are required to make the OSS volunteers to participate actively.

Danesh and Ahmad (2011) in an empirical study on ‘software release planning challenges in software development’ observed that software development is a complicated process and requires careful planning to produce high quality software. Release planning, in large software development projects, may involve a lot of unique challenges and in view of time, budget and some other constraints, there are many problems that may possibly occur. The project managers have been trying to identify and understand release planning, challenges and possible resolutions which might help them in developing more effective and successful software products.

It was found that developers were challenged with release planning as to when to release the next release of software. Concern has been expressed on time scheduling and amount of time allocated to him to finish a task in OSSD. The setting of time for release planning can be of fixed intervals or flexible ones. For some projects, this time is fixed and pre-determined and in others, it is flexible or based on new demands or the condition of the project. Project monitoring was also perceived as a challenge. Complexity of the system lies in delay or problems caused in large projects for delivering a new release. This complexity can be innate and is usually seen in all large software projects. Although most project complexities cannot be totally eliminated, they can only be reduced. Often, technical constraints can also cause complexity which involves a number of technical issues and obstacles which have impact the new release of OSS products. The main finding of the study
was that release activities can be smooth if software defects were identified and suitable features for the new release were added.

Tsay et al (2011) conducted a different type of detailed study on open source software releases, using various methods of versioning, archiving, announcing and publishing the release. Releases are considered as a critical part of the software project. There is no reason to deny the fact that the release history of a software project can shed light on the project history, as well as the release process used by the project, and how those processes change. However, documenting their experiences in Mining Open Source Release Histories, the researchers focused on many release factors which make automating the retrieval of release history information difficult, such as the multiple sources of data, lack of relevant standards and a disparity of tools used to create releases.

As per the researcher’s experiences, the nature of open source software lends itself to data mining and analysis. Although comprehensive datasets of open source projects exist, they generally do not include release history information; and such datasets contain source control, bug tracker, and mailing list statistics, but not release information. In fact, Release information is usually much more complex to obtain, as it is not stored in any standardized format between OSS projects, or even within projects to a large extent. Further, the release process changes, the method of recording the data by volunteers may also undergo change.

As the OSS dataset matures and becomes more comprehensive, they predicted the release activities, as well as the collection mechanism for further use by the OSS community.

Danesh et al (2011) investigated the role efficacy in the release management of OSSD in the context of quality of the product and the
satisfaction of the customers to a great extent. The research investigated the degree of challenges in release management which are effective role players within software domain. Eleven small software companies in Malaysia were chosen as sample and interview constituted the tool for collecting the relevant data. The important findings of the study include the following: Management of different types of releases is different. A major release usually introduces new capabilities, functions or some new and significant changes, and minor releases incorporate a number of fixes for known problems into the baseline, updated to the existing current release. A release which is developed after one year and getting feedback from various customers is different from a release with some minor defects only.

Researchers concluded that release planning is a systematic approach for managing the releases. Using a proper tool for release management in a software organization means an acceptable approach for release management and the right tool to some extent is related to automation of release management.

OSSD needs its own specific software release model, since in comparison to the traditional proprietary software development, it is a different process. Highlighting the significance of the OSS research, Feller and Fitzgerald (2000) described OSS as a typically developed approach by online volunteer communities of programmers and are available to the computer users for downloading, usage, modification and upgradation.

As per the author, according to a recent Gartner survey, by year 2010, 80% of the businesses would have considered using OSS projects and 20% would be using OSS projects in their business transactions. These interests in OSS projects are not just because of their free availability, but also because of their high quality and ability to fulfill user requirements.
2.2 CONCLUSION

It is obvious to note that several researchers have shown keen interest and conducted research on open source software creation, functions and the problems in its operating environment. But none of the study is focused on studying the impact of release management on quality improvement in open source software Project Management. Hence, this was identified as research gap.

In order to fulfill the gap, the researcher turned this gap as his research topic and conducted the study as explorative study, hence, this study.