

Impact of Development Processes and Tools on Success of OSS

T07705



A Thesis Presented to

**Department of Software Engineering
Faculty of Basic & Applied Sciences**

In Partial Fulfillment

of the requirement for the degree

Of

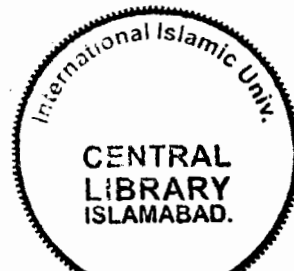
Master of Science (Software Engineering)

By

Zulqarnain

(124-FAS-MSSE/F-06)

**International Islamic University Islamabad
(2011)**



Accession No TH 7705

MS
005.3
ZUI

A handwritten signature in black ink, appearing to be 'S. H. G.' with a large loop on the left side.

1. Open Source software.
2. Computer software Development.

International Islamic University, Islamabad
Faculty of Basic & Applied Sciences
Department of Software Engineering

Dated: 13 April 2011

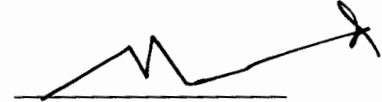
FINAL APPROVAL

It is certified that we have read the thesis, entitled “**Impact of Development Processes and Tools on Success of OSS**”, submitted by Zulqarnain Reg. No. 124-FAS/MSSE/F06 .It is our judgment that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University Islamabad for MS Degree in Software Engineering.

PROJECT EVALUATION COMMITTEE

Supervisor:

Dr. Naveed Ikram
Chairman/Associate Professor
Department of Software Engineering
Faculty of Basic and Applied Sciences
International Islamic University, Islamabad



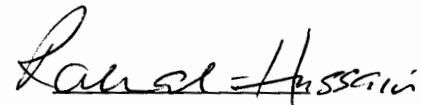
Internal Examiner:

Dr. Abdul Rauf
Assistant Professor
Department of Software Engineering
Faculty of Basic and Applied Sciences
International Islamic University, Islamabad



External Examiner:

Dr. Rahat Bukhari
Director
Department of Computer Science
Quaid-i-Azam University, Islamabad



Abstract

Over the years, the Open Source Software (OSS) development has matured and strengthened, building on some established methodologies and tools. An understanding of the current state of the practice, however, is still lacking. This study presents the results of a survey of the OSS developer community with a view to gain insight OSS development practices of peer review, testing and release management practices, along with the current tool sets used for testing, debugging, build and release management. Such an insight is important to appreciate the obstacles to overcome to introduce certification and more rigour into the development process. It is hoped that the results of this study will initiate a useful discussion and allow the community to identify further process improvement opportunities for producing better quality software.

Acknowledgment

I would like to thank Dr. Naveed Ikram, Chairman, Department of Software Engineering, I.I.U. Islamabad for his continued support and guidance for the completion of this work. He was always available to help and resolve the research issues. His leadership skills helped me to complete the work which was pending since long.

I am equally indebted to Dr. Siraj A. Shaikh, Department of Computing and the Digital Environment, Faculty of Engineering and Computing, Coventry University, United Kingdom for his continued support and professional guidance in the research study.

I am also grateful to the colleagues working in Open Source communities who provided me the necessary knowledge and information relevant to my study.

I am also grateful to Shahida Bibi at International Islamic University for her assistance with data collection for this study.

I would like to thank all of those people who helped make this dissertation possible.

Last but not the least my family whose continued moral support helped me completing this long journey.

Declaration

I hereby declare and affirm that this thesis neither as a whole nor as part thereof has been copied out from any source. It is further declared that I have completed this thesis entirely on the basis of my personal effort, made under the sincere guidance of our supervisor. If any part of this report is proven to be copied out or found to be a reproduction of some other, we shall stand by the consequences. No portion of the work presented in this report has been submitted in support of an application for other degree or qualification of this or any other University or Institute of learning.

Zulqarnain Hashmi

124-FAS/MSSE/F-06

Dedication

I would like to dedicate my work to

ALMIGHTY ALLAH (S.W.T.),

Who is the biggest source of Knowledge,
Who has always showered His endless blessings upon me;

The Teacher for the whole world

Prophet Muhammad (P.B.U.H.),

I also dedicate this work to my family and friends

Whose sincere prayers and love were a source of strength for me
and made this thesis successful.

Table of Contents

1	- Introduction	15
1.1	Open Source Software Overview	15
1.2	Why Open Source Software?	15
1.3	OSS Development Life Cycle	16
1.4	OSS Development Vs Traditional Development	16
1.5	Tools for OSS Projects	17
1.6	Research Focus and Research Question	17
1.7	Research Method	18
1.8	Expected Outcome	18
1.9	Organization of Thesis	19
2	- Literature Review	20
2.1	Open Source Software Development Processes Practices	20
2.2	Management and Requirements Analysis	20
2.3	Roles of OSS Developer and Level of Participation	20
2.4	Peer Review	22
2.5	Testing	23
2.6	Release Management	24
2.7	Tools	26
2.8	Quality and Measures of OSS Project Success	27
3	- Methodology and Procedure	28
3.1	Research Design	28
3.2	Research Approach	28
3.3	Descriptive Research	28
3.4	Data Collection Methods	28
3.5	Population and Sample	29
3.6	Data Collection Tool and Questionnaire Preparation	29
3.7	Pilot Testing of the Questionnaire	30
3.8	Survey Setup and Execution	30
3.9	Data Analysis	31
4	- Research Finding	32
4.1	Developer and Project Profile	32

4.1.1	Development Experience _____	32
4.1.2	Academic Degree _____	33
4.1.3	Job Responsibility _____	33
4.1.4	Level of Participation _____	34
4.1.5	Website Provision _____	35
4.1.6	Activities Provided on Project Website _____	35
4.1.7	Communication Resources _____	36
4.1.8	Authority to Commit Code _____	37
4.1.9	Procedure for Controlling Changes in Software and Documentation _____	38
4.2	Peer Review _____	38
4.2.1	Processes Performed in OSS projects _____	38
4.2.2	Schedule of Peer Reviews _____	39
4.2.3	Frequency of Code Review before it is committed to the code base _____	40
4.2.4	Frequency of Code Review before product release _____	40
4.2.5	Schedule of Peer Review of other people's code in OSS project _____	41
4.2.6	Peer Review Checklist _____	41
4.3	Testing _____	42
4.3.1	Responsible Authority for Testing _____	42
4.3.2	Testing Strategies _____	43
4.3.3	Unit Testing Techniques _____	43
4.3.4	Testing Techniques _____	45
4.3.5	Documented Test Cases _____	46
4.3.6	Formal Testing Procedures _____	46
4.3.7	Schedule of Testing _____	46
4.3.8	Statistical record of testing process for future use and analysis _____	47
4.4	Release Management _____	47
4.4.1	Project Release Authority _____	47
4.4.2	Frequency of Project Releases _____	48
4.4.3	Schedule Strategy for the Project Release _____	48
4.4.4	Criteria of Final Release _____	49
4.4.5	Pre-release Testing Techniques _____	50
4.5	Tools _____	51
4.5.1	Version Control _____	51
4.5.2	Issue Tracking _____	52

4.5.3	Testing tools	52
4.5.4	Peer Review	52
4.5.5	Build System	52
4.5.6	Object Relation Mapping	53
4.5.7	Documentation System	53
4.5.8	Integrated Development Environment	53
5	- Analysis and Conclusion	55
5.1	Analysis	55
5.2	Key Finding	56
5.3	Conclusion	57
5.4	Future work	57
6	- References	58

Appendix A:	Questionnaire
Appendix B:	List of OSS Projects
Appendix C:	Respondent's Responses Sample
Appendix D:	Publication

Glossary

OSS:	Open Source Software
OSSD:	Open Source Software Development
FLOSS:	Free/libre Open Source Software
CMS:	Content management system
CRM:	Customer relationship management
ERP:	Enterprise resource planning
FSI:	Free Software Foundation
OSI:	Open Source Initiative
GPL	General Public License
LGPL:	Lesser General Public License
XML:	Extensible Markup Language
LXR:	Linux Cross-Reference
IRC:	Internet Relay Chat
CVS:	Concurrent Versions System
Git:	A distributed revision control system
Darcs:	Advanced Revision Control System
KDE:	K Desktop Environment
SVN:	Subversion
GUI:	Graphical user interface
GNOME:	GNU Network Object Model Environment
API:	Application Programming Interface
PHP:	Pre Hypertext Preprocessor
IDE:	Integrated development environment
Perl:	Practical Extraction and Reporting Language

- GNU:** Gnu's Not Unix
- FreeBSD:** Berkeley Software Distribution (free Unix-like operating system)
- TDD:** Test Driven Development
- SPSS:** Statistical Package for the Social Sciences

List of Figures

Figure 1: Development experience.....	33
Figure 2: Job responsibility	34
Figure 3: Activities provided on project website	36
Figure 4: Communication resources.....	37
Figure 5: Authority to commit code.....	38
Figure 6: Processes performed in OSS projects	39
Figure 7: Schedule of peer reviews	40
Figure 8: Responsible authority for testing	42
Figure 9: Testing strategies used in OSS project.....	43
Figure 10: Techniques of Unit Testing.....	44
Figure 11: Testing techniques used for OSS	45
Figure 12: Schedule of testing in OSS project	46
Figure 13: Criteria of Final release.....	50
Figure 14: Techniques of pre-release testing.....	51

List of Tables

Table 1: Development experience.....	32
Table 2: Academic degree.....	33
Table 3: Job responsibility	34
Table 4: Level of participation	35
Table 5: Website provision.....	35
Table 6: Activities provided on project website.....	36
Table 7: Communication resources	37
Table 8: Authority to commit code	37
Table 9: Procedure for controlling changes in Software and documentation.....	38
Table 10: Processes performed in OSS projects.....	39
Table 11: Schedule of peer reviews	39
Table 12: Frequency of code peer review before it is committed to the code base.....	40
Table 13: Frequency of code peer review before product release.....	41
Table 14: Schedule of peer review of other people's code in OSS project.....	41
Table 15: Usage of a checklist for peer review	41
Table 16: Responsible authority for testing	42
Table 17: Testing strategies used in OSS project.....	43
Table 18: Techniques of Unit Testing.....	44
Table 19: Testing techniques used for OSS	45
Table 20: Documented test cases used for testing.....	46
Table 21: Usage of formal testing procedures.....	46
Table 22: Schedule of testing in OSS project.....	47
Table 23: Statistical record or reports of testing process for future use and analysis	47
Table 24: Project Release Authority	48
Table 25: Frequency of project releases	48
Table 26: Schedule strategy for the project release.....	49

Table 27: Criteria of final release.....49

Table 28: Techniques of pre-release testing.....50

Chapter 1
Introduction

1 – Introduction

1.1 Open Source Software Overview

Open Source Software (OSS) is becoming popular both in business communities and academic sectors. Free software and OSS has proved its worth as well established alternative to commercial solutions in different software domains with notable products such as *GNU/Linux, Apache, GNOME, KDE, MySQL, PostgreSQL, PHP, Perl, Python, Mozilla, OpenOffice, Eclipse* and even enterprise systems like *openCRX* and *Adempiere*, to name a few.

OSS means software that has not only freely available source code but open access to source code allowing its users to modify and customize the software to better fit their needs. Furthermore, an open source software users can repackage their own version of the software and sell it under certain license.

Open Source term has evolved or introduced three concepts such as Licensing scheme model, Open Source development model and Open Source community [1]. Open source licensing model allows users to access, execute, modify and redistributed source code freely or sell under different business model. This licensing model has established software development model where several developers collaborate asynchronously in a distributed environment across the world. Open source development may usually be initiated by a small group or individual with a project leader to initiate the development task or in the case of organization like Apache or Mozilla take pace with this project leader to initiate these tasks [2]. This development model raised social movement of participants with diverse motivation called the Open Source community.

1.2 Why Open Source Software?

OSS has attracted an increased number of stakeholders as some companies adopt it to reduce development effort, while others for cost saving and flexible solutions. Popularity and importance of OSS can be seen from industry usage as there are many organizations like sun Microsystems, Hewlett-Packard and IBM have started to look towards OSS and OSS

Development Processes as a mean to minimize their development efforts by reuse of open source code, and to provide great flexibility in their development practices [3, 4]. These organizations are even developing open source development laboratory to promote open source software collaboration and growth [5]. OSS is now a major social movement involving an estimated more than 800,000 developers around the world [6]. OSS provides a useful learning platform for software developers and students as it provides a unique environment in which learners can be quickly exposed to real-world innovation.

1.3 OSS Development Life Cycle

The open source software process model and its development processes are emerging as a new paradigm for software development and have already produced several successful products. These development processes are fundamentally different from traditional analysis, design, implementation and testing driven processes.

A general process of Open Source Software Development (OSSD) life cycle can be described from a pendulum model [7]. In Pendulum Model the process begins from requirement analysis and specification, followed by implementation and testing in parallel. The third stage is of validation during which bugs are reported while fourth stage is component submission during which tested component is submitted and hence a new version of the product becomes available, while at the same time new features request is submitted which sets the pendulum to start a new rotation of process.

1.4 OSS Development Vs Traditional Development

OSS development is typically initiated by a small group of people [2] and can be distinguished from traditional development mentioned as under [7, 8]:-

1. Open source software developed by usually large number of volunteers while traditional development process involves a dedicated team
2. Open source Software's features are build by volunteers on the basis of their need and interest. No formal requirement engineering process is used in OSS. While in traditional software development market demands provide basis for software features and formal requirement engineering is followed.

3. Volunteer choose what they want to work on. No work is assigned by a leader or organization in OSSD. While in traditional software development work is assigned to team members.
4. Less development tool supported as compare of traditional software development
5. There is no system or detailed design document in OSSD, while in traditional software development formal design phase is followed before programming.
6. Improper project development plan and schedules in OSSD, while in traditional software development proper project planning and schedule is followed.
7. There is no list of deliverables in OSSD compared to traditional software development
8. There is unstructured testing and quality assurance in OSSD compared to traditional software development where formal testing and quality assurance methodologies are used.

1.5 Tools for OSS Projects

In early years a set of tools consisting of mailing list, bugs text file, an install text file and CVS server are commonly used during OSS project development. Now a day's use of other various tools for version control, issue tracking, technical support, build system, design and code generation, quality assurance and documentation generation is also common [9].

Some tools in OSS development still not well supported for activities like requirements management, project management, metrics, estimation, scheduling, and test suite design. The unavailability of such tool is however justifiable because open source projects do not need to meet deadlines or balance budgets [9, 10].

1.6 Research Focus and Research Question

Our research effort is aimed at better understanding of the development processes and behavior within a set of OSS projects. To identify important development processes i.e. peer review, testing, release management and tools in OSS especially in web based application domain, we have gone through quantitative analysis of successful Open source web based projects.

OSS projects have been criticized for lack of clear and open detail of development processes. OSS projects even Apache, Mozilla and Net Beans usually do not provide much clear and open detail of development processes. Most of the studies of OSS projects like Apache and Mozilla [11, 12, 13] usually give an informal description of development processes which cannot be reused in other projects by OSS developers.

Observation discovers that information sources i.e. community information, project history, work roles and task description provided on several OSS project websites appear mind-numbing and ambiguous [15]. A need for standard and clear development practices has been acknowledged [14, 17, 20]. Sharing clear and open description of development processes, with a view to further modify and reuse is certainly of great interest to the wider OSS community [13, 17]. This will definitely allow OSS developers to identify process improvement opportunities to produce higher quality software and more successful OSS [13, 17].

This study addresses the research question: **“What development processes and tools practices are being used in successful OSS projects?”**

1.7 Research Method

This study is based on survey of OSS developers. The survey is essentially descriptive in nature and lies in cross-sectional time dimensional category. The top 250 OSS projects from a variety of domains were selected from SourceForge [18] and Launchpad [19] on the basis of downloading ratings.

The sampling ensured that each member of the population has an equal probability of being selected. Download rates do not convey quality or success but certainly offers a measure of fitness for purpose as users of OSS have actively downloaded it [17]; it is essentially an objective measure independent of our influence.

1.8 Expected Outcome

This study may identify commonly in use development processes and tools for better quality and success of OSS. The importance of quality in OSS due to development processes has already been acknowledged [14, 17, 20]. The outcome of this research may provide guidelines or key finding that may help open source communities to select proper

development processes to enhance quality of OSS. Such an insight is important to appreciate the obstacles to overcome to introduce certification and more rigors into the development and testing processes.

1.9 Organization of Thesis

The rest of this thesis is organized as follows. Chapter 2 describes some of the related work. Some past observations and the relevant trends observed are brought to attention. Chapter 3 discusses the methodology adopted for this effort with particular emphasis on the choice of OSS projects targeted for the survey. This is helpful in setting the results in the overall context. Chapter 4 presents the research finding of the survey. Some trends of interest are highlighted though the majority of the results serve to affirm traditional perceptions of the OSS community. Chapter 5 concludes this study and highlights some future work.

Chapter 2
Literature Review

2 – Literature Review

2.1 Open Source Software Development Processes Practices

Processes are vibrant interaction among OSS developers working on OSS projects. OSS development is attracting more organization to check on the possibilities of more profits that could be generated by engaging researchers of high standing [3]. This chapter highlights the outcome with regard to management, requirement analysis, roles, level of commitment, peer review, testing and release management available in literature.

2.2 Management and Requirements Analysis

Past research reflect that there is no continue formal planning, project management and requirement analysis in OSS. A point validly raised is that the latest software engineering methods are usually not adopted by OSS [9, 22]. Virtual management is something more relied by the OSS projects. Users get involved into management field at their will. OSS is also believed as it is not interested to get into planning stage. However, projects often do have TODO lists that form a sort of agenda for development [23]. Sacchi states that OSS lacks and do not possess conventional documents [22]. Instead requirements are found through emails which are the result of arguments between users and developers. Such requirements may point out what is required and what not. Mockus [12] says that users can expect its needs through bug complaints and suggestions.

2.3 Roles of OSS Developer and Level of Participation

Organizational structures, technical roles and career opportunities within the OSS community have been widely studied [24, 25, 26]. Traditionally software engineers have been restricted to roles like requirements analyst, software designer, programmer, or code tester. Such individuals move up towards the higher authority in a project whereas in the OSS community, roles and progression (or movement) is more sundry: volunteering roles can move up and down amongst different paths much more gracefully, with the possibility of lateral movements as well.

A role can be complex in different contexts. For example a participant can have a role of 'core developer' in one context while in another context he can have a role of 'support person' [26, 27].

Some of the recognized roles in OSS include *project leader, core developer or member, active developer, passive or peripheral developer, bug fixer, bug reporter, reader/active user* and *passive user*, with the likely possibility of overlap. Not all of these types of roles exist in all OSS communities, and some communities may use different names. For example, some communities refer to core members as maintainers. The difference between bug fixer and peripheral developer is also rather small as peripheral developers are likely to be engaged in fixing bugs [26].

Lin conducted interviews and survey of developers who move from community involvement to working for a company but still doing OSSD, they have 30 hybrid roles such that they draw on resources from the company and community while balancing their loyalties [28].

Many individuals work as volunteers in OSS projects. Hence their work on OSS projects is not their main job but appears to be kind of part time activity. Even if they are paid for their participation, the case remains same. In Apache all core participants had other main jobs which typically use the software for which they were volunteer contributors [29].

Balance between work time and spare time is necessary. Apache contributors are reported to spend 30% of their work time on web servers [30]. Ability to spend long uninterrupted hours on OSS projects is reported to be a major benefit [31].

As the projects get matured, a significant growth of communities results in to large user communities and developer groups [32]. Contribution of developers is mainly driven by personal or community needs; but a higher commercial motivation also exists in large projects. Excellent level of development knowledge can be observed in community which indicates that developers know the right way of approaching the things. This explains the success of OSS projects even with loose collaborations, informal processes and fewer rules.

Diverse sizes of core and peripheral groups exist for different projects. For the bug tracking system, distribution of contribution was studied for 120 Sourceforge teams [33]. The statistics showed that developers list on Sourceforge was not a good indicator of core membership and

that contribution in communication field was higher than code field. Core group size was measured 5% of all participants of the project [33].

Fraction of contribution of different developers and community members is different in OSS projects. Mockus observed that in Apache community top 15 contributors (out of 388) contributed 66% of problem reports and over 83% of modification requests, which is quite high. Comparison of these contribution distributions to commercial projects suggest that although OSS projects have high contributors, most of the activity remains centralized especially for new features [12].

2.4 Peer Review

Several developer and user examining the source code is one of the fundamental principles that underlie OSS. A well-accepted speech that “given enough eyeballs, all bugs are shallow” has confined that principle [39]. Some Source code of OS application is constantly watched by developers and users; from minute quantity to a large scale. Some reports show that peer review on OSS has been performed by millions of developers [38]. A survey indicates that only 50% of respondents reported that their code was reviewed and only 9% of OSS developers claimed the peer review of the entire source code [43]. One explanation given for these statistics is that code might be reviewed later as it was open for inspection by any interested group.

Peer reviews are mostly done before and after any source code is committed to the repository [40]. Peer review has many similarities with distributed, asynchronous reviews. They are certainly more extensively acknowledged and accepted as part of the organizational culture in OSS than in traditional development. The developers more likely to perform them without any directions, which although not vital, may be a sign of commitment to quality within OSS projects. It is useful in detecting flaws, defects and quality of OSS, and is well recognized in software engineering generally for its crucial role [40]. Peer review is more important part of the organizational culture for OSS development as compared to traditional development [41]. The developer’s attitude towards peer reviews can be an indication of an OSS organization’s commitment to quality.

Evidences also figure out that peer review is also performed by most of the non-OSS project developers [43]. It is also reported that a team of four to six inspectors perform peer reviews.

Institute of Electronic and Engineers suggested standard for conducting software reviews is three to six persons [44]. Among three reviewers one must be accountable to the effectiveness of the assessment. Software inspections are reported to detect and eliminate faults more cheaply as compared to testing [45].

Although the team members vary but the main emphasis is to maximize the ability to find bugs. The actual task is classified to be either ad-hoc or based on some checklist. The former signifies that the reviewer's team has to examine in a perfect manners to dig out imperfections without any guidance. In order to guide and facilitate reviewers in examining all defect forms, standardized checklist of frequent faults is considered. There is good evidence that checklist-based techniques tend to find more defects than ad-hoc techniques [46].

2.5 Testing

Recent studies established uniqueness of OSS model by showing the exceptionally high user involvement, structured approaches for flaws/bugs handling process, considerable section of testing in software development life cycle and great usage of configuration and bug tracking tools. However, deficiency in documentation, design documentation and complexity of maintained source code causes the limited access for developers during testing [32].

Unit testing is the most frequent in OSS development [40]. Pre-release testing on broader perspectives is less common, with the idea being that the released candidate is dealt with by the users and its flaws reported.

Pre-release testing is not commonly demonstrated and formal testing is even not implemented for most of the OSS development [40, 27]. With confidence in code peer reviewed, many OSS developers are content with only minor testing [42]. Some other sources go as far as to claim that over 80% of OSS developers do not have any plans of testing [43].

It is accepted by majority of the OSS developers that if someone downloads the code then he would also progress to examine and report bugs. Thus OSS developers rarely perform testing and depend on others to point out defects [43].

There is no specific evidence with regards to usage of automated tools but debuggers are widely acknowledged [43]. For regression tests, about 48% of OSS projects follow baseline testing, whereas proportion is relatively higher in mega projects [47]. A study conducted on the Apache project revealed that no system testing or regression was performed [12]. Further analysis reports that while regression test suites were available for Apache they were not actually mandatory [48]. This complements with suggestions that improvement is required in quality assurance practices, applied processes and project success criteria [32].

Different testing procedures are followed in OSS development. Linux kernel release process does not have any formal testing procedure [49]. Developers test their code before submission and peer review is done by users. While formal testing procedures are followed in projects like Linux Test Project, Linux stabilization Project [50]. FreeBSD is also reported to have a formal testing process [51].

2.6 Release Management

Release management is a vital part in OSS development. OSS development has varied approaches for release which results into availability of latest version, which sometimes is stable and sometimes not. Michlmayar [53] presents a comparative study on release management to find that it can be categorized into three types, with respect to the concerned audience and the effort required to deliver the release: *developer release* for interested developers and experienced users requiring less or no effort, *major or stabilized releases* for end users requiring more effort to deliver with considerable new features and functionality, bugs fixed and tested, and *minor releases or updates* for existing users requiring a slight effort for stabilized release.

More generally, a feature-based strategy is adopted in which certain criteria or goals have to be fulfilled, or, a time-based strategy with particular dates set for release and used as orientation for release. Time based strategy is more preferred [52].

A release manager is mainly responsible of delivery of high quality software to user. In small and large scale project, release managers have same responsibilities, such as delivering a stable release, but very different role. For example, in small scale projects they have administrative role (release preparation in different format, release notes formation, software

distribution) but in large scale projects they have coordinative role (making sure that different parts of software is ready all together, developer are focused for stable release). Small project often faces human resource problem whereas large project deals with more fundamental problems, such as coordination with teams for creating a stabilized release.

Different tools are tightly integrated with overall development process and with release management, such as version control and bug tracking system. Same as particular practices also allied to release management like freezing the development to concentrate on faults removal and release publication, scheduling specially in projects which adopts time based strategy; ascertaining goals; setting the cut-off date; building for different platform; user testing; following a release list by making it sure that all necessary steps are followed; and holding a post release review.

The most common problems faced in release management process are unprepared main features, interdependencies among different parts of project or among projects, no or little support for old releases and less interest in user releases by the developers as they themselves use developer version. Sometimes because of delay vendors start shipping development releases which causes fragmentation, less testing of modifications causes major delays, coordination problems among individual developers and stabilizing their work at one fell swoop [53].

Apache httpd, Subversion and Linux were compared for different dimensions of release procedure i.e. release authority, versioning, prerelease testing, and approval of releases, distribution, and formats. Substantial variations were found among the projects [48].

Some OSS projects adopt informal release schedules. As in Linux releases come at irregular schedules [49]. The reason of release schedule was not obvious. However, release procedure was followed by first accepting code patches for some time and then freezing acceptance of any new patches to let the system get stabilized. Stabilization was evaluated only by absence of reported problems. However, some projects have more formal and organized procedure of releasing. FreeBSD has a regular schedule of release i.e. four months [51]. FreeBSD also has a release team to coordinate release and follows a release pattern for incremental release of projects.

2.7 Tools

In early years OSS projects just used normal set of tools consist of mailing list, bugs text file, an install text file and CVS server. Nowadays there are number of tools used in OSS projects for version control, issue tracking, technical support, build system, design and code generation, quality assurance and documentation generation.

The following tools are reported to be used in OSS projects [9, 10].

- **Version Control System**

Concurrent Versions System (CVS) and subversion used as version control system or source control. WinCVS, MacCVS, TortoiseCVS, CVSWeb, and ViewCVS are CVS clients whereas RapidSVN and TortoiseSVN are used as Subversion clients.

- **Issue Tracking**

Bugzilla and Scarab are bug/Issue tracking system.

- **Build System**

Most commonly build system are Make, Automake, and Autoconf. 'Make' UNIX command is a standard tool to automate the compilation of source code trees. Ant is a Java replacement for 'make' that uses XML based build files. Tinderbox, Gump, CruiseControl, Xenofarm, and Maven are nightly build tools automatically compile a project's source code and produce error report.

- **Object Relation Mapping**

Author[10] has identified OSS developers have implemented Object Relation Mapping tools like Torque, Castor, and Hibernate. They are using at the same rate that traditional developers have.

- **Documentation System**

XDoclet, vDoclet, JUnitDoclet, and Doxygen are javadoc API documentation code generation tools have been used in OSS.

- **Testing Tools**

Quality Assurance tools such as JUnit, PHPUnit, PyUnit, and NUnit are mainly used for unit testing whereas Codestriker is used for code reviews.

For Mozilla project, software engineering processes of Mozilla and tools used are related to each other [13]. These tools include Bugzilla, a web application for tracking bugs, reviewing and correction; Bonsai, a tool which is used to query CVS code repository; LXR, a hypertext based source code browser; and Tinderbox, an automated build and regression system.

2.8 Quality and Measures of OSS Project Success

Quality significantly affects the usage of OSS projects and enhances OSS success [34, 35]. Quality of OSS projects is defined from different notations i.e. *quality by access*, *quality by development* and *quality by design*. These notations describe the quality emphasis on different stages of software life cycle [37]. It is reported that quality control approach depends on developer's commitment to the process.

Numerous measures have been proposed to evaluate the success of OSS. Seven proposed measures for OSS project success are; system and information quality, user satisfaction, usage, individual and organizational impacts, project output, process and outcomes for project members [36].

Aberdour has suggested several important processes for quality OSS development. These include *sustainable community*, *code modularity*, *co-evolvement of system and community*, *rapid release cycles*, *code review*, *project team environment* and *a mixed type of testing including formal and OSSD testing* [8]. This shows that OSS success can be defined from different angles. Among these measures system and information quality is reported to be most frequently used. Although code quality and documentation quality are reported, important measures as well in theory for OSS. A link between success of OSS projects and maturity of process has been identified although kind of relationship is yet unidentified [17].

Chapter 3

Methodology and Procedure

3 – Methodology and Procedure

3.1 Research Design

This study intends to provide insight of development processes and tool being used in open source projects. This chapter describes the research approach, the development of instrument for gathering the information, identification of target population and sampling procedure, design of procedure for information collection, the way to collect research data and gives a description on how the data may be analyzed.

3.2 Research Approach

Research method to carry out this study is survey which is the most common method for generating primary data. The type of survey is descriptive in nature. Research is laid in Cross-Sectional time dimensional category. Unit of analysis is individual and OSS developers are the respondent of survey.

3.3 Descriptive Research

Descriptive research is the exploration of the existing certain phenomena or want to gain a better understanding of a topic like to find out the detail of trends that would not be known. This study is also descriptive research that describes current practices of development methodologies and tools applied in OSS.

3.4 Data Collection Methods

Beside survey there are also other ways to collect data like conducting interviews, content analysis from the respective project web site or other recourses but in case of OSS development mostly projects have insufficient information regarding their development processes on wikis and websites. Interviews from OSS developers are also difficult to conduct because of their less availability in Pakistan. So on line survey is the best suited data collection method of this study.

3.5 Population and Sample

The concept of population is vital as it defines the set of entities from which the research sample is to be drawn. The population of interest in this study is Open source developers. The targeted population is developers of recognized OSS initiatives that have good download rating. The most important source to collect information about the development processes and tools used in OSS projects is OSS communities such as those accessed through *SourceForge* and *LaunchPad*, where thousands of OSS projects are hosted across several domains. Our selected domains are *business intelligence and performance management, digital archiving, CMS systems, CRM, e-commerce, ERP, email client, frameworks, message boards, project management, scheduling, site management, social networking, ticketing systems* and *wiki*. We selected the top 250 OSS projects from these domains on the basis of downloading ratings from SourceForge [18] and Launchpad [19]. SourceForge is currently the largest hosting site for free and OSS projects and also largest number of OSS developers are registered there [15]. Sample selection is an important procedure of survey, systematic sampling method was chosen because in which each member of the population has an equal probability of being selected. Note that we use the download rate to define success. Downloads do not convey quality or success of OSS but certainly offers a measure of fitness for purpose as users of OSS have actively downloaded it. Downloads do provide the advantage as measure as it is objective and dependent on the users [17].

TH 7705

3.6 Data Collection Tool and Questionnaire Preparation

Questionnaires seem to be the appropriate way to gather information from the OSS developers across the world. We have designed online questionnaire (Appendix A) consist of 33 questions, divided into five groups: OSS developer profile, OSS project profile, Peer Review, Testing, Release management and Tools. The main objective of the questionnaire is to determine the development processes and developmental tools being used in OSS projects. The initial questions were created and reviewed by my supervisor. Some Peer review and testing questions drew inspiration from [16] and rest from literature review.

3.7 Pilot Testing of the Questionnaire

Validity and reliability are the main priorities in surveys and Pilot testing is a technique to achieve these goals. Without pilot testing questionnaire are not valid and reliable. To ensure content validity, the content of the questions addressed the research questions and main objectives of the study. There is a need of pilot testing to assess the questionnaire clarity, understandability, comprehensiveness and acceptability. We validated our questionnaires by faculty members and OSS industry experts and their reliability was determined by getting few responses from the population. Participants were given an opportunity to offer comments on the structure of the questions including clarity, relevance to the objectives of the study, level of difficulty and length of the survey. After pilot testing, several changes were made to the questionnaire as per the feedback.

3.8 Survey Setup and Execution

A detailed search was undertaken to identify projects which existed with the same name under different domains of SourceForge [18] and Launchpad [19]. 250 projects were identified after eliminating duplications. Once the list of OSS projects was decided, names and contact details of the respective developers were collected from their hosting websites using PHP based custom web crawler. Some developers were also involved in more than one project, which were also excluded for duplication. We restricted answers in our questionnaire for a specific project.

The survey was developed using LimeSurvey. LimeSurvey is one of best and tested Open Source survey software and can be compared with existing good featured commercial survey software. LimeSurvey software was configured and hosted under my own domain <http://www.lissomsurvey.com>.

The survey was a closed survey where e-tokens were automatically generated and emailed to the concerned OS developer to access survey. Unique token was received by prospective respondent. At the beginning of the survey form, the participants were informed that their responses would be used only for the purpose of this research and their identity would not be shared. They were told that filling out this survey would take approximately 15 to 20 minutes.

Respondents received e-mails explaining the goals and purposes of the survey and asking their contribution. After the first email, reminder e-mails were also sent over some time to non-respondents only. The data was collected automatically in MySQL database after survey form submission. Some questions had kept optional for the ease of respondent.

As questionnaire was divided into five groups, survey was also divided into five sections. Respondents were allowed to move previous section and review their answers any time before final questionnaire submission. Respondents only had one opportunity to submit the questionnaire with their e-token, after which that token was disabled. The individual response rate was 13.3%, and the project response rate was 35.5%.

3.9 Data Analysis

Data analysis is an important phase of the descriptive research study. A comparative data analysis strategy has been chosen to analyze the data. Limesurvey software has the ability to export data into statistical software's format like SPSS and MS Excel. SPSS software was used to analyze descriptive statistics and chi square test. The chi square test is used to determine whether an association between two categorical variables in a sample is likely to reflect a real association between these two variables in the population. One-Way Classification is used in chi square to find out the number of responses that fall in other categories.

Chapter 4
Research Finding

4 – Research Finding

This section presents the research finding of our survey. Section 4.1 discusses the profile of projects and individuals who responded to the survey. This sets the context for the following sections which delve into peer review practices in Section 4.2, testing strategies in Section 4.3, release management in Section 4.4 and the use of tools in Section 4.5. Section 4.6 provides a brief analysis on the results commenting on the aspects that are of particular interest.

4.1 Developer and Project Profile

4.1.1 Development Experience

Over 58% of the total developers surveyed have more than 5 years of experience working with OSS. Of the rest, over 18% have 3 to 5 years, over 17% have 1 to 3 years and just under 5% of the developers have less than one year of experience working with OSS. Over 3% preferred not to answer (Figure 1).

Less than 1 year	4	4.71%
1-3 years	15	17.65%
3- 5 years	16	18.82%
5+ years	50	58.82%

Table 1: Development experience

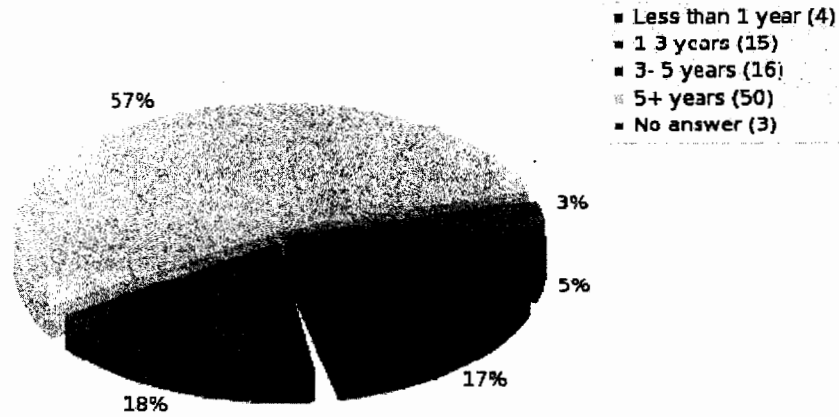


Figure 1: Development experience

4.1.2 Academic Degree

Over 36% of the total developers who responded claimed a graduate degree, with over 25% holding a masters degree and just under 12% holding a doctoral degree (Table 2). Other recorded responses are high school, self-educated, engineering degree & B-tech. Even some developers have no academic degree. Around 20% respondent couldn't answer for any reason.

Academic Degree	Count	Percentage
Bachelor's degree	31	36.47%
Master's degree	22	25.88%
Doctoral degree	10	11.76%
Other	10	11.76%

Table 2: Academic degree

4.1.3 Job Responsibility

Job Responsibility	Count	Percentage
Project leader	27	31.76%
Project Manager	6	7.06%

Core Developer	22	25.88%
Active Developer	9	10.59%
Passive Developer	8	9.41%
Bug Reporter	3	3.53%
Bug Fixer	0	0.00%
Other	10	11.76%

Table 3: Job responsibility

Over 31% of the total respondents identified themselves as project leader, over 25% as core developer, over 10% as active developer and over 9% as passive developer. Just over 7% claimed project management and over 3% bug reporter roles. Just under 12% fell in the *other* category, which included translator, localizer and community manager roles (Figure 2).

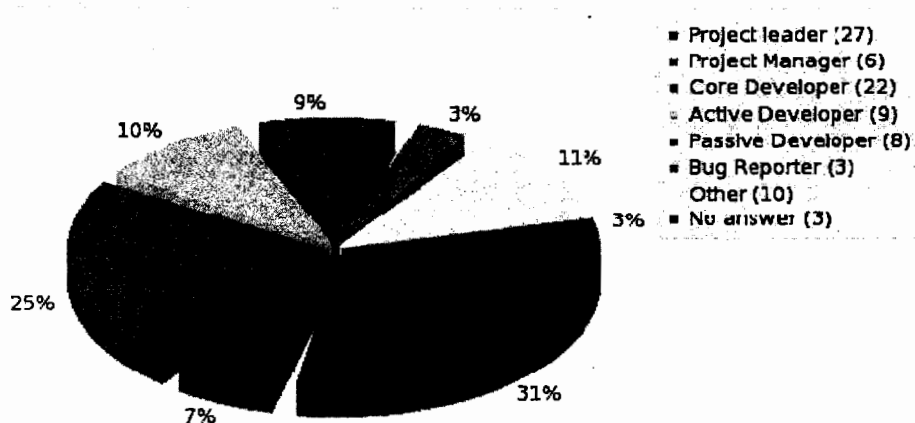


Figure 2: Job responsibility

4.1.4 Level of Participation

Dedicated full-time	21	24.71%
----------------------------	----	--------

Part-time	52	61.18%
Other	12	14.12%

Table 4: Level of participation

Out of the total respondents, over 61% of the developers participate only part-time participation whereas over 24% are as dedicated full-time. A small percentage, just over 14%, described their participation as either in *free time*, *voluntarily* or *occasional*.

4.1.5 Website Provision

Mostly sampled successful projects have their own dedicated web site.

Yes	82	96.47%
No	3	3.53%

Table 5: Website provision

4.1.6 Activities Provided on Project Website

When asked about information provision and dissemination for their OSS projects, over 96% of the respondents claimed that their project has a dedicated website. Over 91% identified announcements, over 87% provide some form of user documentation and just over 81% mentioned a feature list advertised for the project. Mailing lists, tutorials and to-do lists were also identified by well over 50% of the respondents. Some other avenues for communications identified included code collaborator, repositories, forums and case studies provided for the users (Figure 3).

Announcements	78	91.76%
Feature List	69	81.18%
To-Do list	45	52.94%

Mailing list	56	65.88%
User Documentation	74	87.06%
Developer Documentation	59	69.41%
Tutorials	51	60.00%
Other	22	25.88%

Table 6: Activities provided on project website

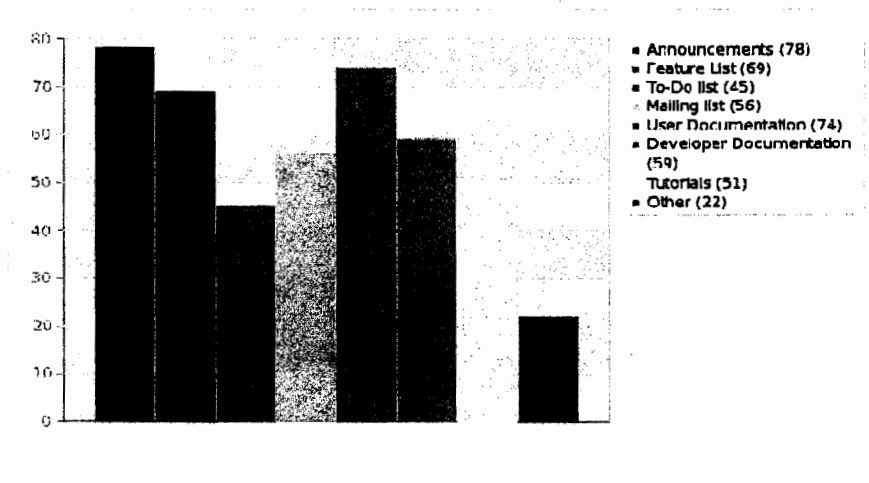


Figure 3: Activities provided on project website

4.1.7 Communication Resources

For internal communication a variety of resources were identified including mailing lists (by over 76%), threaded discussion forums (60%), IRC/chat/instant messaging (over 55%), newsgroups (over 17%), community digests (just under 13%) and other resources such as XMPP, bug trackers, wiki sites and micro blogging (over 16%).

Threaded discussion forums	51	60.00%
Mailing List	65	76.47%
Newsgroups	15	17.65%
IRChat/Instant messages	47	55.29%

Community digests	11	12.94%
Other	14	16.47%

Table 7: Communication resources

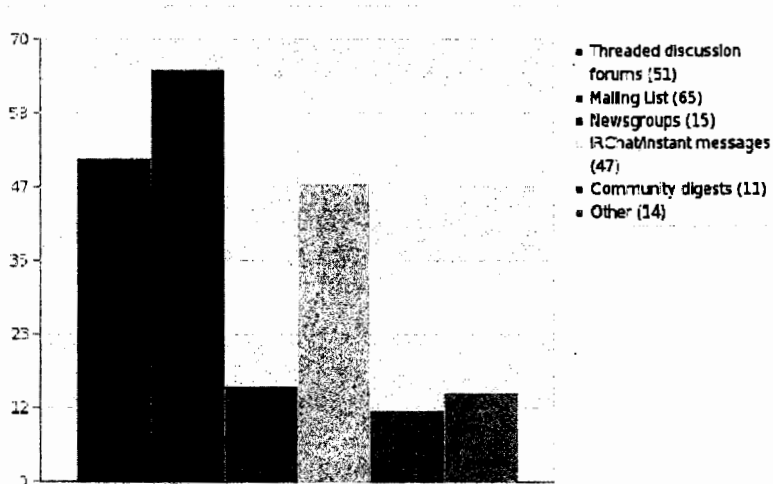


Figure 4: Communication resources

4.1.8 Authority to Commit Code

The authority to commit code varies from project to project, with the majority allowing core developers (just under 92%) to commit. Over 60% mentioned active developers and over a quarter mentioned passive developers with the ability to commit.

Core Developer	78	91.76%
Active Developer	53	62.35%
Passive Developer	22	25.88%
Other	8	9.41%

Table 8: Authority to commit code

Others who commit code in project are Contributors, all developer, dedicated localizer, admin and QC.

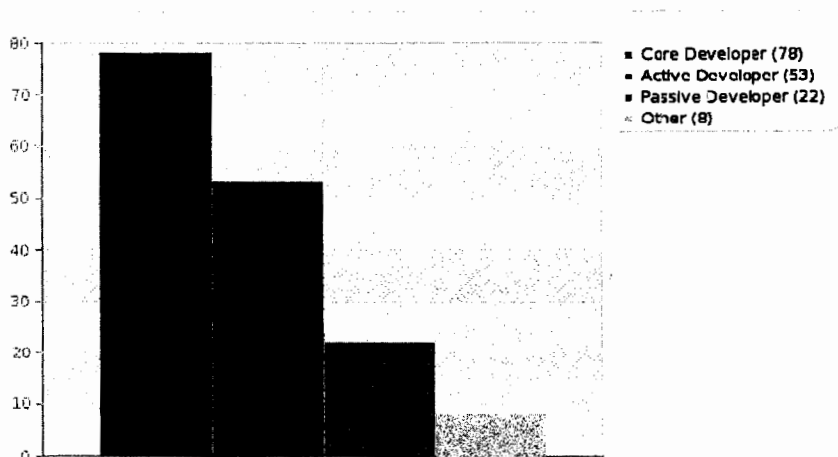


Figure 5: Authority to commit code

4.1.9 Procedure for Controlling Changes in Software and Documentation

Response	Count	Percentage
Yes	58	68.24%
No	10	11.76%

Table 9: Procedure for controlling changes in Software and documentation

4.2 Peer Review

4.2.1 Processes Performed in OSS projects

Process	Count	Percentage
Peer Reviews	52	61.18%
Testing	74	87.06%
Release Management	74	87.06%

Table 10: Processes performed in OSS projects

The survey reveals that software testing and release management are far more prevalent than code review, with over 87% confirming that some form of testing and release management is carried out on the OSS project they are involved in. Only over 61% of the respondents claimed any code review for the software they are involved with. This is somewhat surprising as almost 40% of the respondents did not claim any code review on their projects.

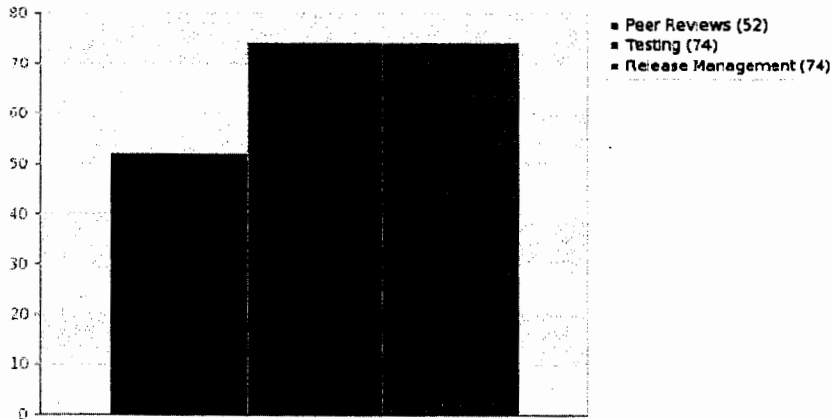


Figure 6: Processes performed in OSS projects

4.2.2 Schedule of Peer Reviews

Before any source code is committed to the code base (continuously)	31	36.47%
Before product release	24	28.24%
Randomly	26	30.59%
Never	7	8.24%
Other	10	11.76%

Table 11: Schedule of peer reviews

Of those who did affirm code review, over a third claimed that reviews are performed before

any source code is committed to the code base. Around 30% also confirmed that some review is performed randomly and before product release.

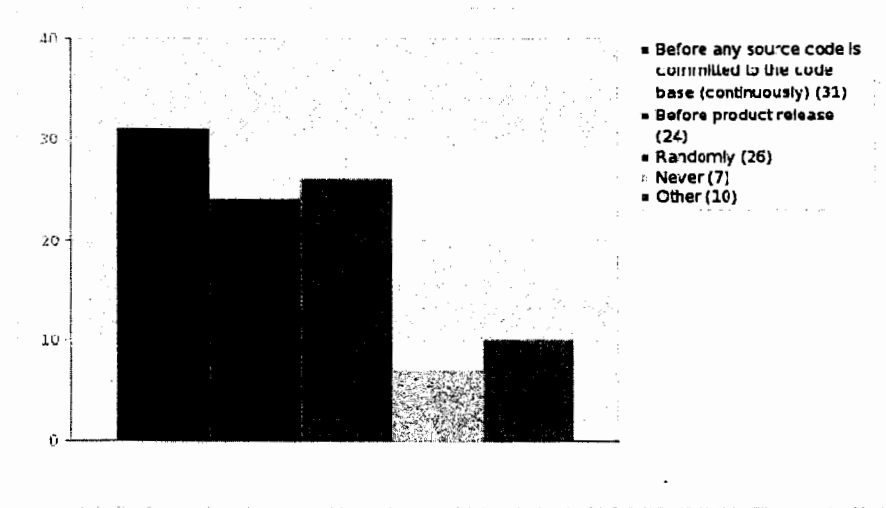


Figure 7: Schedule of peer reviews

4.2.3 Frequency of Code Review before it is committed to the code base

Frequency of Code Review	Count	Percentage
Always	27	31.76%
Quarterly	1	1.18%
Half the time	4	4.71%
Occasionally	16	18.82%
Never	13	15.29%

Table 12: Frequency of code peer review before it is committed to the code base

4.2.4 Frequency of Code Review before product release

Frequency of Code Review	Count	Percentage
Always	27	31.76%
Quarterly	2	2.35%

Half the time	4	4.71%
Occasionally	15	17.65%
Never	10	11.76%

Table 13: Frequency of code peer review before product release

4.2.5 Schedule of Peer Review of other people’s code in OSS project

Always	22	25.88%
Quarterly	4	4.71%
Half the time	7	8.24%
Occasionally	26	30.59%
Never	8	9.41%

Table 14: Schedule of peer review of other people’s code in OSS project

An important element of code review is inspection of code written by others. Over a quarter of those surveyed affirmed that they regularly review other’s code with just over 30% claiming occasional review of other’s code.

Only under 10% said they have never reviewed source code written by others. This reflects very well highlighting a strong ethos of evaluation and self regulation amongst the section of the OSS community.

4.2.6 Peer Review Checklist

Yes	9	10.59%
No	46	54.12%

Table 15: Usage of a checklist for peer review

4.3 Testing

4.3.1 Responsible Authority for Testing

Developers	79	92.94%
Q&A team	23	27.06%
Passive Users	44	51.76%
None	1	1.18%
Other	7	8.24%

Table 16: Responsible authority for testing

Other responsible for testing are merge manager, core users, and community members.

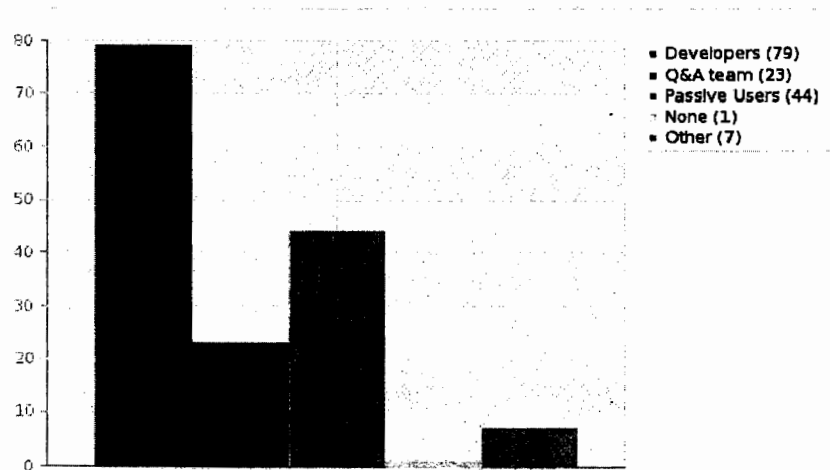


Figure 8: Responsible authority for testing

There is strong evidence that developers have the primary responsibility for testing according to the 93% of the respondents. Testing is also left to users on over half of the projects. Dedicated individuals for quality and assurance are also identified by over 27% of the respondents.

4.3.2 Testing Strategies

Testing Strategy	Count	Percentage
Black Box Approach	38	44.71%
White Box Approach	34	40.00%
None	12	14.12%
Other	12	14.12%

Table 17: Testing strategies used in OSS project

The type of testing carried out is of interest here: nearly 45% of the respondents identified a black box approach to testing, with a similar 40% identifying a white box approach. Other strategies recorded from responses are TDD, DUnit, test suites, partial test plan, test for errors, build, and review in use.

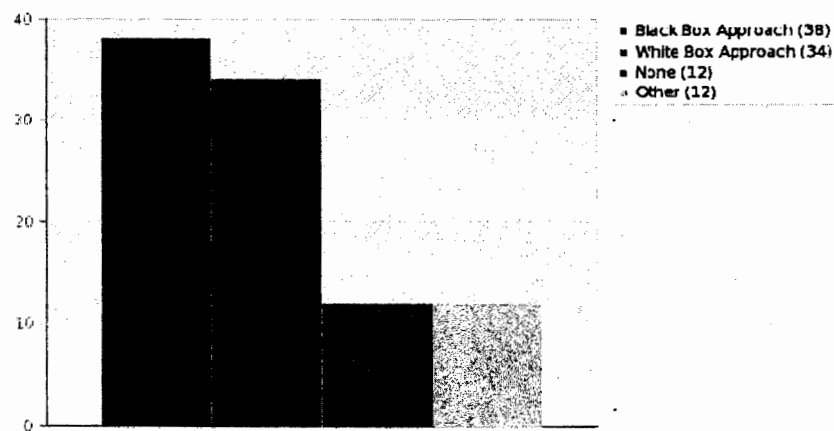


Figure 9: Testing strategies used in OSS project

4.3.3 Unit Testing Techniques

Unit Testing Technique	Count	Percentage
Statement Testing	30	35.29%
Loop Testing	15	17.65%

Path/Branch Testing	18	21.18%
Mutation Testing	5	5.88%
None	19	22.35%
Other	8	9.41%

Table 18: Techniques of Unit Testing

For unit tests, over 35% of the developers mentioned statement testing, over 21% mentioned path/branch testing, over 17% mentioned loop testing and just under 6% claiming mutation testing. Responses also pointed out other unit testing techniques which are result testing, test driven development, regression tests and predefined test cycle.

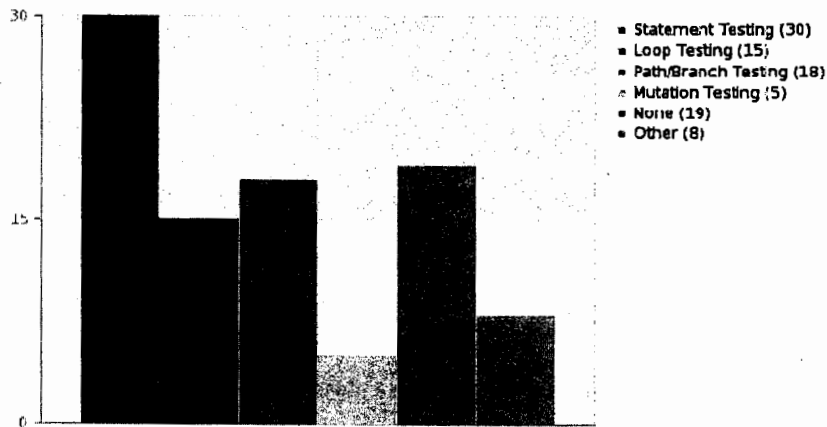


Figure 10: Techniques of Unit Testing

4.3.4 Testing Techniques

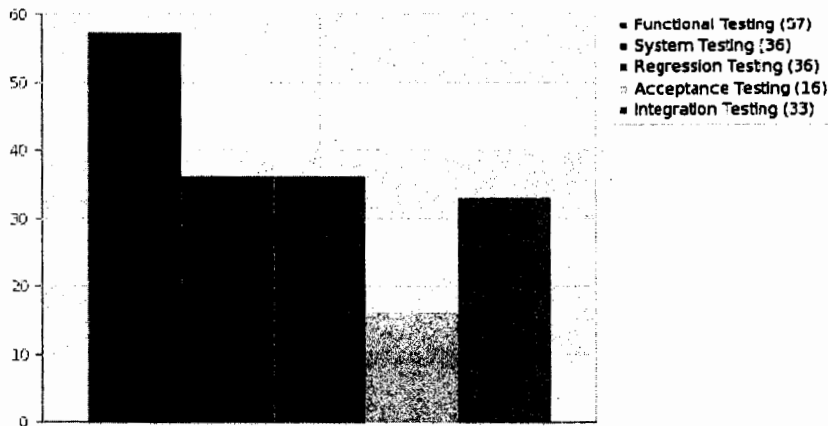


Figure 11: Testing techniques used for OSS

A range of testing techniques are adopted by OSS projects. When offered to identify multiple techniques, survey respondents affirmed functional testing is adopted by over 67% of the projects, with some form of system testing by over 42%, regression testing by over 42%, integration testing by just under 39% and acceptance testing by under 19%.

Testing Technique	Number of Projects	Percentage
Functional Testing	57	67.06%
System Testing	36	42.35%
Regression Testing	36	42.35%
Acceptance Testing	16	18.82%
Integration Testing	33	38.82%

Table 19: Testing techniques used for OSS

4.3.5 Documented Test Cases

Response	Count	Percentage
Yes	42	49.41%
No	28	32.94%

Table 20: Documented test cases used for testing

4.3.6 Formal Testing Procedures

Response	Count	Percentage
Yes	36	42.35%
No	37	43.53%

Table 21: Usage of formal testing procedures

Over 42% of the projects are said to have some formal testing procedure, while more than 43% are said to have no formal testing procedure.

4.3.7 Schedule of Testing

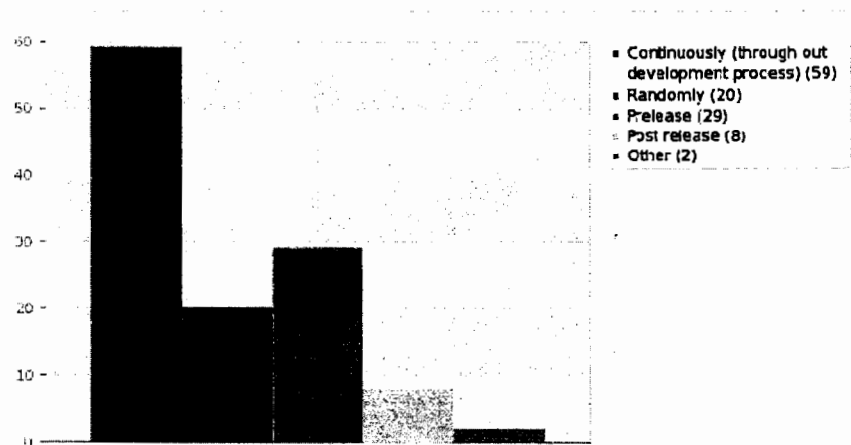


Figure 12: Schedule of testing in OSS project

Over two-thirds of the projects affirmed a continuous schedule for testing, with over a third also claiming pre-release testing. Post-release testing was also highlighted by around 10% of the projects.

Testing Schedule	Count	Percentage
Continuously (throughout development process)	59	69.41%
Randomly	20	23.53%
Pre release	29	34.12%
Post release	8	9.41%
Other	2	2.35%

Table 22: Schedule of testing in OSS project

4.3.8 Statistical record of testing process for future use and analysis

Response	Count	Percentage
Yes	21	24.71%
No	45	52.94%

Table 23: Statistical record or reports of testing process for future use and analysis

Only under a quarter of the projects keep any form of statistical testing for future use and analysis.

4.4 Release Management

4.4.1 Project Release Authority

Authority Type	Count	Percentage
Project Leaders	48	56.47%
Dedicated Manager	10	11.76%
Elected Candidate	3	3.53%

Core Developer	15	17.65%
Community Developer	3	3.53%
Other	3	3.53%

Table 24: Project Release Authority

Clear and consistent release procedures are important if an OSS project is to provide a coordinated and timely delivery. Our survey reveals over half of the project leaders to have complete release authority with under 20% of the projects also allowing core developers to have authority over release. Some projects also identified dedicated release or product managers having release authority. Other release authorities recorded are Release manager, Product Manager and Committee recorded from other response.

4.4.2 Frequency of Project Releases

Every month	9	10.59%
Every quarter	10	11.76%
Every six months	25	29.41%
Every year	9	10.59%
Other	21	24.71%

Table 25. Frequency of project releases

30% release every six months, with 11% releasing every quarter and a similar percentage every year. Other recorded information shows that it depends on developments, monthly to quarterly, Randomly, Every 9 to 12 months.

4.4.3 Schedule Strategy for the Project Release

Fixed dates	13	15.29%
Fixed features	8	9.41%

Release when its ready	42	49.41%
Release often and early	12	14.12%
Other	2	2.35%

Table 26: Schedule strategy for the project release

Nearly half of the projects release when ready, with just over 15% releasing on fixed dates, a similar number releasing often and early and only under 10% releasing for fixed features.

4.4.4 Criteria of Final Release

Committer's consensus	16	18.82%
Zero-Bug Reporting in beta release	15	17.65%
Core team consensus	47	55.29%
Market demand (Community)	20	23.53%
Manger decision	26	30.59%
Fixed Date	10	11.76%
Other	3	3.53%

Table 27: Criteria of final release

Over 55% of our respondents affirmed that core team consensus is the basis for release. Almost a third also rely on single release authority's decision, with a similar number also citing market demands, committers' consensus and zero bug reporting in beta release also as contributing factors. Other responses show that judgment of product manager also affects the final release.

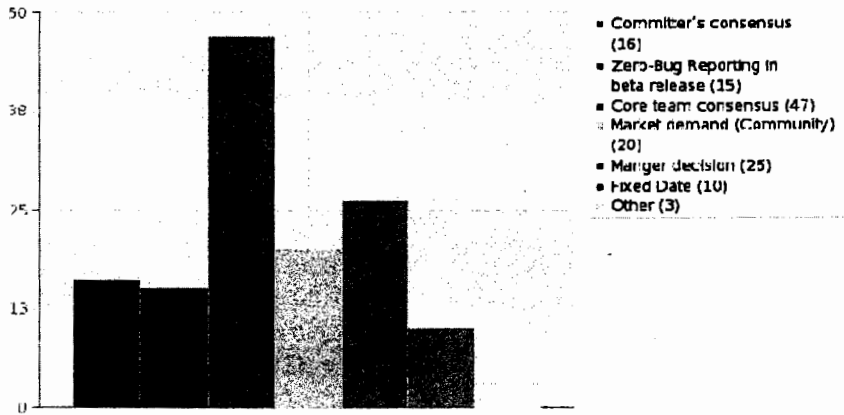


Figure 13: Criteria of Final release

4.4.5 Pre-release Testing Techniques

Technique	Count	Percentage
Regression Test suite	34	40.00%
Smoke Test	15	17.65%
Unit Tests	35	41.18%
System Test	26	30.59%
Other	10	11.76%

Table 28: Techniques of pre-release testing

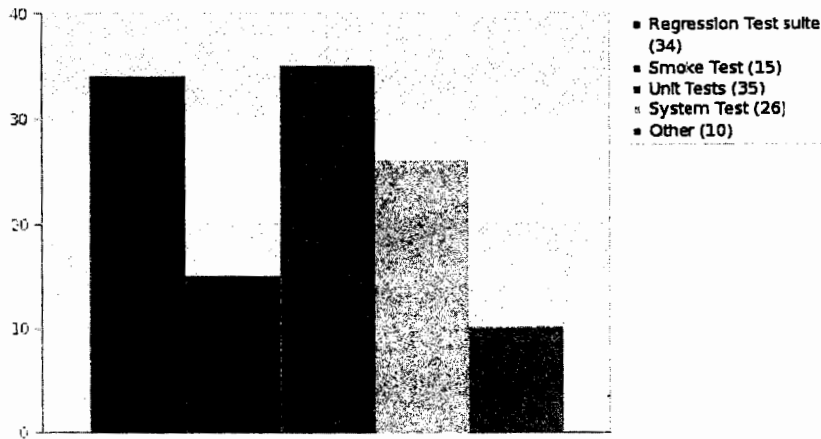


Figure 14: Techniques of pre-release testing

4.5 Tools

In this section the most commonly tools identified by the survey respondents are highlighted. This is helpful as it offers some insight into the choice of tools for OSS.

4.5.1 Version Control

Version control systems are undoubtedly crucial for OSS development as they allow management of changes to source code and documents. Our survey reveals Subversion as the most common version control system used, followed by Git and CVS. Some other choices are Mercurial, Bazaar and Darcs. Only Git and Mercurial are distributed systems. With a distributed version control system there isn't one centralized code base to pull the code from repository. Different branches hold different parts of the code. Other version control systems, such as SVN and CVS, use centralized version control.

TortoiseSVN is the most popular client for those who have affirmed the use of Subversion. RapidSVN, Textmate SVN and KDESvn are some of the other clients identified. KDESvn is a graphical client for the Subversion version control system on the KDE desktop. Subclipse is using as add on providing support for Subversion within the Eclipse IDE. AnkhSVN and VisualSVN are Subversion clients, implemented as a source control plug in for Microsoft Visual Studio. SvnX is an open source GUI subversion client for Mac. msysGit is using as

Git client for windows. *GitX* is a Git GUI made for Mac OS. TortoiseCVS, WinCVS and Cygwin are using as CVS client. *TortoiseHg* and *hgweb* are the clients for Mercurial version control system. Darcs client for Darcs version control system. Bzr-Eclipse is the plugin that provides Bazaar integration support in the Eclipse and bzt-gtk is a Bazaar plugin to provide support in GNOME desktop environment.

4.5.2 Issue Tracking

Issue tracking systems allow individual or groups of developers to keep track of outstanding bugs or issues effectively. Mantis, Bugzilla and Trac are the most popular issue tracking systems identified in our survey. Issue trackers provided by Sourceforge, Google, Codeplex and Launchpad are also identified. Other similar systems mentioned include JIRA, FogBugz, Roundup, Zentrack and YouTrack, demonstrating a very wide variety of systems in use.

4.5.3 Testing tools

A huge variety of tools supporting testing are identified in our survey including JUnit, easymock, PHPUnit, CTest, DUnit, Litmus, nosetests, Python unittest, QUnit, Selenium, Hudson, buildbot, NUnit, MsTests, ReSharper, TestDriven, NCover, Zope Unit testing, Ruby unit test, Squish (Froglogic), NUnit, MbUnit, GNU autotools, Pootle, scalacheck, Maven Invoker Plugin, MyTAP and GTest. There is no clear pattern for a single most popular tool, perhaps due to the nature of the activity involved.

4.5.4 Peer Review

Smart Bear Code Collaborator, Fisheye, Bugzilla, Eclipse and Pootle are some of the most popular tools identified.

Some developers are also using hosting services like GitHub, Launchpad and version control system client like Mercurial as a tool for code reviews. Some projects have also built custom tool for code review like extension to mediawiki and others.

4.5.5 Build System

A wide variety of build systems are identified including Ant, Make, Automake, CMake, Gnu Autotools, Tinderbox, Hudson, Bamboo, NAnt, MsBuild, Maven, SBT (Scala-based Simple

Build Tool), XCode, Python setuptools, Buildout, buildbot, Module::Build, Rake (Ruby), TeamCity, PEAR (PHP Extension and Application Repository) and Eclipse.

4.5.6 Object Relation Mapping

Embarcadero, SQLAlchemy, .NET ORM, XOOps ORM, NHibernate, ActiveRecord and Eclipse are tools that generate persistence layer for their application. Some projects have created their own custom tools for ORM.

4.5.7 Documentation System

The most common documentation system identified in the survey is Doxygen, which offers support for both on-line and off-line documentation from a set of source files. It supports many programming languages like C++, C, Java, Objective-C, Python, PHP and C#. Other tools identified include Epydoc and Sphinx (for generating API documentation for Python), Javadoc (for generating API documentation in HTML format from source code), Sandcastle, DocProject, DelphiCodeToDoc, phpDocumentor (phpDoc) and RDoc.

Sandcastle and DocProject are open and extensible system for authoring, managing and building compiled documentation solution for Microsoft platform. DelphiCodeToDoc is a free documentation system for Delphi.

phpDocumentor or phpDoc is a tool that can be used to create documentation from php source files. RDoc generates documentation from Ruby Source files. Open Architecture Community System (OpenACS) is an open source web application framework that has integrated support for generating documentation.

4.5.8 Integrated Development Environment

Eclipse has been recognized as the most common development platform amongst the community surveyed.

Following is the list of other IDE used in OSS development.

- CodeLite is an open-source, cross platform IDE for the C/C++ and other programming languages
- VisualStudio and ReSharper for Dot Net based application development

- Quanta HTML editor and a web development for the K Desktop Environment
- TextMate is a general-purpose GUI text editor for Mac OS X
- Kate is a text editor for KDE
- Delphi and Lazarus is a cross platform visual IDE which provides a Delphi mimic development environment
- Komodo IDE, the cross-platform tool for teams working in all scripting languages
- Notepad++ is a free source code editor that supports several multi programming languages like Perl, Python, C++ and PHP
- Qt Creator is a cross-platform application and UI framework by Nokia
- Vim is also cross-platform test editor supporting all major scripting languages
- Emacs is an extensible, customizable text editor by GNU
- Xcode is the development tool that is integrated with the Cocoa and Cocoa Touch frameworks for building Mac and iPhone apps
- NetBeans is full featured java based IDE
- Eclipse-Pydev is a Python IDE for Eclipse
- PyPaPi internal written IDE for python

Chapter 5

Analysis and Conclusion

5 – Analysis and Conclusion

5.1 Analysis

With over half the respondents having over 5 years of experience with OSS development, our survey is informed by an extensively experienced group of individuals. With nearly a two-third of the community contribution being as part-time, this reflects on the voluntary yet dedicated nature of participation by the sampled OSS community. A majority of the respondents were either project leaders or core developers with a graduate degree.

Needless to say, most projects claim to have some procedure in place for controlling changes to software and supporting document. Most of them allow core developers to commit code, where as nearly two-third also allowing active developers to commit. This is critical because it implies that any significant changes that need to be brought in to improve developmental processes would not only require consent on behalf of the core developers of the project but also depend on their adoption of new practice as well.

When it comes to testing, unit testing is most common with a strong focus on functional testing. Nearly, 50% of the projects are also using some form of documented test cases. This sends a strong hint as to where more rigour and assurance measures could be incorporated in OSS development in general. Strict and specific testing for critical functionality could be the key here to associate any standard evaluation of the software and any certification that may follow. Note that projects that have adopted some formal testing procedure are also the ones where release management is an integral part of the project.

It is interesting to observe that nearly all OSS projects use a wide range of communication tools and strategies with nearly all having a dedicated website. Feature lists, mailing lists, user and developer documentation are other most common mechanisms in use. This demonstrates the need for effective and efficient communication that the disparate set of users employ to contribute to the success of OSS.

For the purposes of change of developmental practices and adoption of more rigorous means, our research findings offer to identify a starting point. It is the experienced members of the community that are best placed to bring about this change. This may appear counterintuitive as developers who are in set their ways are least likely to be agents of change. The results, however, reveal that it is indeed the most experienced (those with over 5 years of experience) of developers who perform peer review, and are responsible for testing on their projects, and have the ability to commit code and authority to release. Of those with lesser experience, a very small proportion falls in this category.

5.2 Key Finding

Following is the summary of key finding of this research study.

- OSS development is still depend on part time participation
- Announcements, feature list, mailing list, documentation, forum and instant messaging are most important activities performed on dedicated project web sites
- Core and active developers have the permission to commit code in their official repository
- There are procedures to control changes to software and supporting documents
- Testing and release management are integral processes of OSS projects where as peer review is performed partially in OSS projects
- Normally peer review is performed before source code is merged to code base
- No checklist practice is used in peer review process
- Developers and passive users actively participated in testing
- Normally testing is performed continually throughout the project, developers and passive users actively participate in this process
- Common testing technique used is unit testing and functional testing
- Only half of the developers are using documented test cases and formal testing procedures in successful OSS projects
- No trend to keep statistical record of testing for future use and analysis
- Project leader and core developer have the authority to release projects
- Final release is made on the basis of Core team consensus
- Schedule strategy for releasing project depends on when release is ready
- Average project releasing period is approximately 6 month

- Regression and unit testing techniques are normally used for pre-release testing
- Subversion is the most popular version control system and TortoiseSVN is the most common client for subversion
- Mantis, Bugzilla and Trac are the most popular issue tracking systems
- Variant tools are used in testing, peer review and release management
- Doxygen is the most popular documentation system used in OSS projects
- Eclipse is identified as the most common development platform

5.3 Conclusion

This research study may contribute to understand the OSS developer community better and the state of current development practices. It serves to provide a snapshot which is both useful and indicative of further inquiry.

The motivation behind this work follows from earlier work [21] that encourages a more rigorous approach to software development and testing within the OSS community. Any such change therefore has to be brought about carefully. The finding may serve to highlight a prevailing structure of OSS projects, which should be taken advantage of. The leadership for any such initiative should also ideally come from within the community. This will facilitate adoption and better stands to influence the younger and future generations of developers who are to follow.

5.4 Future work

The target population for this survey has provided with a rich sample of the community some of whom could be targeted for further inquiry. Further follow-up survey can explore the perceptions of formal methods and more rigorous methods alike for adoption by the community. Aspects of software modeling and verification, assurance and certification can be explored.

6 – References

- [1] K. Crowston, K. Wei, J. Howison, A. Wiggins. Free/libre open source software development: What we know and what we do not know. *ACM Computing Surveys*, 2010.
- [2] W. Scacchi, J. Feller, B. Fitzgerald, S. Hissam, K. Lakhani. Understanding Free/Open Source Software Development Processes. *Software Process: Improvement and Practice* 11(2):95–105, 2006.
- [3] A. W. Brown, G. Booch. Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors, *Proceedings of the 7th International Conference on Software Reuse. Lecture Notes in Computer Science*, 2319:123-136, 2002.
- [4] J. Lonchamp. Open Source Software Development Process Modeling; *International Series in Software Engineering*, Springer, 10:29-64, 2005.
- [5] S. Sharma, V. Sugumaran, B. Rajagopalan. A framework for creating hybrid-open source software communities. *Information Systems Journal* 12 (1):7-25, 2002.
- [6] R. M. Stallman, L. Lessig, J. Gay. Free Software, Free Society: Selected Essays of Richard M. Stallman. *GNU Press*, 2002.
- [7] V. Potdar, E. Chang. Open Source and Closed Source Software Development Methodologies, *Collaboration, Conflict and Control, Proceedings of the 4th Workshop on Open Source Software Engineering*, pp. 105-109, 2004.
- [8] M. Aberdour, Achieving Quality in open source, *IEEE computer society*, 2007.
- [9] J. Robbins, H. Fitzgerald, S. Lakhani. Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools. *Perspectives on Free and Open Source Software*, pp. 245–264, 2005.

- [10] J. E. Robbins. Adopting OSS methods by adopting OSS tools. *2nd Workshop on Open Source Software Engineering (co-located with 24th International Conference on Software Engineering) Orlando, Florida, 2002.*
- [11] A. Mockus, R. Fielding, J. Herbsleb. A Case Study of Open Source Software Development: The Apache Server. *Proceedings of the 22nd International Conference on Software Engineering (ICSE), Los Angeles, CA, pp. 263–272, 1999.*
- [12] A. Mockus, R. Fielding, J. D. Herbsleb. Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology* 11(3):309–346, 2002.
- [13] C. R. Reis, R. P. de Mattos Fortes. An overview of the software engineering process and tools in the mozilla project. *Workshop on OSS Development, Newcastle upon Tyne, UK, pp. 162–182, 2002.*
- [14] W. Scacchi. Issues and Experiences in Modeling Open Source Software Development Processes. In *In Proceedings of the 3rd ICSE workshop on Open Source Software Engineering*. Pp. 121–125. 2003.
- [15] J. Howison, K. Crowston. The perils and pitfalls of mining SourceForge, *Proceedings of the International Workshop on Mining Software Repositories*, pp. 7–11, 2004.
- [16] G. Koru, K. E. Emam, A. Neisa, M. Umarji. A Survey of Quality Assurance Practices in Biomedical Open Source Software Projects. *Journal of Medical Internet Research* 9(2):e8, May 2007.
- [17] M. Michlmayr. Software Process Maturity and the Success of Free Software Projects. *Software Engineering: Evolution and Emerging Technologies* 130:3–14, 2005.
- [18] SourceForge Home page. <http://sourceforge.net/>, 2009. <http://sourceforge.net/>
- [19] LaunchPad Home page. <https://launchpad.net/>, 2009. <https://launchpad.net/>

- [20] S. A. Shaikh, A. Cerone. Towards a metric for Open Source Software Quality. In Barbosa et al. (eds.), *Foundations and Techniques for Open Source Certification 2009*. Electronic Communication of the European Association of Software Science and Technology (ECEASST) 20. 2009.
- [21] A. Cerone, S. A. Shaikh. Incorporating Formal Methods in the Open Source Software Development Process. In *International Workshops on Foundations and Techniques bringing together Free/Libre Open Source Software and Formal Methods (FLOSS-FM 2008) & 2nd International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2008)*. UNU-IIST Research Report 398, pp. 26–34. 2008.
- [22] W. Scacchi. Free/Open Source Software Development Practices in the Computer Game Community, *IEEE Software*, 56-66, 2004.
- [23] Y. Yamauchi, M. Yokozawa, T. Shinohara, T. Ishida. *Collaboration with lean media: How open-source software succeeds*. Proceedings of the 2000 ACM conference on Computer supported cooperative work, pp. 329-338, 2000.
- [24] W. Scacchi. Free/Open Source Software Development : Recent Research Results and Emerging Opportunities. *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, 2007.
- [25] C. Jensen, W. Scacchi. Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study, *Proceedings of the 29th international conference on Software Engineering*, pp.364-374, 2007.
- [26] Y. Ye, K. Kishida. Toward an understanding of the motivation Open Source Software developers. *Proceedings of the 25th International Conference on Software Engineering*, pp. 364–374, 2003.
- [27] C. Gacek, B. Arief. The many meanings of Open Source. *IEEE Software*, 21(1):34-40, 2004.

- [28] Y. Lin. Hybrid Innovation: How Does the Collaboration Between the FLOSS Community and Corporations Happen? *Knowledge, Technology and Policy*, 18(4): 86-100, 2006.
- [29] R. T. Fielding. Shared leadership in the Apache Project. *Association for Computing Machinery. Communications of the ACM*, 42(4), 1999.
- [30] K. R. Lakhani, E. V. Hippel. How open source software works: "free" user-to-user assistance. *Research Policy*, 32(6): 923-943, 2003.
- [31] K. Crowston, J. Howison, C. Masango, U. Y. Eseryel. *Face-to-face interactions in self-organizing distributed teams*. 2005.
- [32] T. Otte, R. Moreton, H. D. Knoell. Applied Quality Assurance Methods under the Open Source Development Model. *IEEE 32nd International Computer Software and Applications Conference (COMPSAC)*, pp. 1247-1252, 2008.
- [33] K. Crowston, K. Wei, Q. Li, J. Howison. *Core and periphery in Free/Libre and Open Source software team communications*. Proceedings of the 39th Annual Hawaii International Conference on System Sciences, IEEE Computer Society, 2006.
- [34] S. Y. T. Lee, H. W. Kim, S. Gupta, "Measuring open source software success," *Omega*, vol. 37, no. 2, pp. 426-438, April, 2009.
- [35] K. Crowston, H. Annabi, J. Howison, "Defining Open Source Software Project Success", *Proceedings of the 24th International Conference on Information Systems (ICIS)*, Seattle, WA, pp. 327-340, 2003.
- [36] K. Crowston, J. Howison, H. Annabi. Information systems success in Free and Open Source Software development: Theory and measures. *Software Process-Improvement and Practice*, 11(2): 123-148, 2006.
- [37] S. A. Shaikh, A. Cerone. Toward a quality model for open source software (OSS). *Elsevier Science B. V.*, 2007.
- [38] S. Greiner, B. Boskovic, J. Brest, V. Zumer. Security issues in information systems based on open source technologies. *EUROCON*, 2003.

- [39] Eric S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly; 1999.
- [40] T. Halloran, W. Scherlis. High quality and open source software practices. *2nd Workshop on Open Source Software Engineering, International Conference on Software Engineering*, pp. 19–25, 2002.
- [41] K. E. Emam. Software Inspection Best Practices. *Agile Project Management Advisory Service* 2(9), 2001.
- [42] Stark J. Peer reviews as a quality management technique in open-source software development projects. *European Conference on Software Quality 2002*:340-350.
- [43] L. Zhao, S. Elbaum. A survey on quality related activities in open source. *ACM SIGSOFT Software Engineering Notes*, pp. 53–57, 2000.
- [44] STD I. IEEE Standard for Software Reviews and Audits. IEEE STD 1028-1988 1989.
- [45] Ackerman AF, Buchwald LS, Lewski FH. Software inspections: an effective verification process. *IEEE Software*, 6(3):31-36, 1989.
- [46] A. Dunsmore, M. Roper, M. Wood. The development and evaluation of three diverse techniques for object-oriented code inspection. *IEEE transactions on software engineering* 29(8):677–686, 2003.
- [47] L. Zhao, S. Elbaum. Quality assurance under the open source development model. *The Journal of Systems and Software* 66:65–75, 2003.
- [48] J. R. Erenkrantz. Release management within open source projects. *Proceedings of the Third Workshop on Open Source Software Engineering*, 2003.
- [49] D. G. Glance. Release criteria for the Linux kernel. *First Monday*, 9(4), 2004.

- [50] T. Zimmermann, P. Weissgerber, S. Diéhl, A. Zeller. Mining Version Histories to Guide Software Changes. *IEEE Transactions on Software Engineering*, 31(6):429-445, 2005.

- [51] T. T. Dinh-Trong, J. M. Bieman. The FreeBSD project: A replication case study of open source development. *IEEE Transactions on Software Engineering*, 31(6):481-494, 2005.

- [52] M. Michlmayr. Quality Improvement in Volunteer Free and Open Source Software Projects – Exploring the Impact of Release Managemen. PhD's thesis, University of Cambridge, UK, 2007.

- [53] M. Michlmayr, F. Hunt, D. Probert. Release management in free software projects: Practices and problems. *IFIP International Federation for Information Processing, Open Source Development, Adoption and Innovation* 234:295–300, 2007.

Appendix A
Questionnaire

Questionnaires

Section I

1. Please specify your development experience in Open Source Software?

Please choose only one of the following:

- Less than 1 year
- 1-3 years
- 3- 5 years
- 5+ years

2. What is the highest academic degree do you have?

Please choose only one of the following:

- Bachelor's degree
- Master's degree
- Doctoral degree
- Other _____

3. What is your current job responsibility on OSS project?

Please choose only one of the following:

- Project leader
- Project Manager
- Core Developer
- Active Developer
- Passive Developer
- Bug Reporter
- Bug Fixer
- Other _____

4. What level of participation do you have in developing OSS?

Please choose only one of the following:

- Dedicated full-time
- Part-time
- Other _____

5. Please write down the name of your current open source project (Specify only one project name for which you are going to answer) Please write your answer here:

Section II

6. Whether a website is provided for OSS project?

Please choose only one of the following:

- Yes
- No

7. Which of the following information/activities are provided on project web site?

Please choose all that apply:

- Announcements
- Feature List
- To-Do list
- Mailing list
- User Documentation
- Developer Documentation
- Tutorials
- Other _____

8. Which of the following communication resources are available in your OSS project? Please choose all that apply:

- Threaded discussion forums
- Mailing List
- Newsgroups
- IRChat/Instant messages
- Community digests
- Other _____

9. Who has the authority to commit code in project? Please choose all that apply:

- Core Developer
- Active Developer
- Passive Developer
- Other _____

10. Is there a procedure for controlling changes to the software and supporting documents? Please choose only one of the following:

- Yes
- No

Section III

11. Which of the following processes are performed in your OSS project?

Please choose all that apply:

- Peer Reviews
- Testing
- Release Management

12. When peer reviews are performed? Please choose all that apply:

- Before any source code is committed to the code base (continuously)
- Before product release
- Randomly
- Never
- Other _____

13. How often is code peer reviewed before it is committed to the code base?
Please choose only one of the following:

- Always
- Quarterly
- Half the time
- Occasionally
- Never

14. How often is code peer reviewed before product release?
Please choose only one of the following:

- Always
- Quarterly
- Half the time
- Occasionally
- Never

15. How frequently do you peer review other people's code in your OSS project?
Please choose only one of the following:

- Always
- Quarterly
- Half the time
- Occasionally
- Never

16. Do you use a checklist for peer review?

Please choose only one of the following:

- Yes
- No

Section IV

17. Within your project who is responsible for testing?

Please choose all that apply:

- Developers
- Q&A team
- Passive Users
- None
- Other _____

18. Which of the following testing strategies are used in your OSS project?

Please choose all that apply:

- Black Box Approach
- White Box Approach
- None
- Other _____

19. For unit testing which of these techniques are used?

Please choose all that apply:

- Statement Testing
- Loop Testing
- Path/Branch Testing
- Mutation Testing
- None
- Other _____

20. For testing which of these techniques are used for your software?

Please choose all that apply:

- Functional Testing
- System Testing
- Regression Testing
- Acceptance Testing
- Integration Testing

21. Are documented test cases are used for testing?

Please choose only one of the following:

- Yes
- No

22. Do you have any formal testing procedures?

Please choose only one of the following:

- Yes
- No

23. Schedule of testing in your OSS project is

Please choose all that apply:

- Continuously (throughout development process)
- Randomly
- Pre-release
- Post release
- Other _____

24. Do you keep any statistical record or reports of testing process for future use and analysis?

Please choose only one of the following:

- Yes
- No

Section V

25. Who is release authority for project release?

Please choose only one of the following:

- Project Leaders
- Dedicated Manager
- Elected Candidate
- Core Developer
- Community Developer
- Other _____

26. How often the project releases?

Please choose only one of the following:

- Every month
- Every quarter
- Every six months
- Every year
- Other _____

27. Which of the schedule strategy is followed for the Project?

Please choose only one of the following:

- Fixed dates
- Fixed features
- Release when its ready
- Release often and early
- Other _____

28. Final release is made on the basis of

Please choose all that apply:

- Committer's consensus
- Zero-Bug Reporting in beta release
- Core team consensus
- Market demand (Community)
- Manger decision
- Fixed Date
- Other _____

29. Which techniques are used for pre-release testing?

Please choose all that apply:

- Regression Test suite
- Smoke Test
- Unit Tests
- System Test
- Other _____

Section VI

30. Which of the following version control system is used in your project?

Please choose only one of the following:

- CVS
- Subversion
- Other _____

31. Which of the following version control system's client you are using?

Please choose all that apply:

- WinCVS
- TortoiseCVS
- CVSWeb
- ViewCVS
- RapidSVN
- TortoiseSVN
- SmartCVS
- Other _____

32. Which bug/issue tracking system is used in your OSS project?

Please choose all that apply:

- Bugzilla
- Scarab
- Gnats
- JIRA
- Mantis
- FogBugz
- Other _____

33. Write down the name of tools you are using in OSS project?

Please write your answer(s) here:

Testing: _____

Peer Review: _____

Build Process: _____

ORM (Object Relation Mapping): _____

Source code documentation generator: _____

IDE (Integrated Development Environment): _____

34. Please do let us know about your suggestions and comments about this survey

Please write your answer here:

Appendix B

List of OSS Projects

List of OSS Projects

No.	Project Name	Registered	Latest File	Downloads	Members	URL
1	XAMPP	09/04/02	02/20/09	19,426,506	5	http://sourceforge.net/projects/xampp/
2	phpBB	07/06/00	12/13/08	18,737,759	11	http://sourceforge.net/projects/phpbb/
3	phpMyAdmin	03/18/01	02/28/09	18,129,768	11	http://sourceforge.net/projects/phpmyadmin/
4	FreeMind	06/18/00	03/09/09	7,371,793	8	http://sourceforge.net/projects/freemind/
5	AppServ	10/10/01	05/10/08	7,266,806	1	http://sourceforge.net/projects/appserv/
6	XOOPS Web Application Platform	12/07/01	03/25/09	7,187,366	150	http://sourceforge.net/projects/xoops/
7	LimeSurvey	02/21/03	08/20/09	6,400,48	44	https://sourceforge.net/projects/limesurvey
8	DotNetNuke	03/22/03	12/12/08	5,799,598	9	http://sourceforge.net/projects/dnn/
9	FloAt's Mobile Agent	01/09/03	02/03/07	5,311,688	32	http://sourceforge.net/projects/fma/
10	wxWidgets	08/16/00	03/17/09	4,650,786	34	http://sourceforge.net/projects/wxwindows/
11	TYPO3 Content Management Framework	02/12/01	03/11/09	3,887,299	26	http://sourceforge.net/projects/typo3/
12	wxPython	09/01/00	02/20/09	3,246,250	3	http://sourceforge.net/projects/wxpython/
13	SquirrelMail	11/18/99	12/03/08	3,207,681	11	http://sourceforge.net/projects/squirrelmail/
14	GanttProject	01/29/03	12/20/08	2,974,211	6	http://sourceforge.net/projects/ganttproject/
15	TinyMCE	02/27/04	03/26/09	2,470,783	3	http://sourceforge.net/projects/tinymce/
16	Liferay Portal	03/18/02	02/27/09	2,260,250	6	http://sourceforge.net/projects/liferay/
17	eGroupWare: Enterprise Collaboration	04/13/03	11/24/08	2,014,309	4	http://sourceforge.net/projects/egroupware/
18	Pentaho - Business Intelligence	06/01/05	03/31/09	1,930,969	18	http://sourceforge.net/projects/pentaho/
19	Compiere ERP + CRM Business Solution	06/09/01	03/26/09	1,479,029	75	http://sourceforge.net/projects/compiere/
20	MediaWiki	08/25/01	02/20/07	1,457,944	67	http://sourceforge.net/projects/wikipedia/
21	vtiger CRM	08/23/04	11/13/08	1,445,291	32	http://sourceforge.net/projects/vtigercrm/
22	Zen Cart	06/20/03	12/11/07	1,418,329	5	http://sourceforge.net/projects/zencart/
23	PopTray	03/19/03	09/17/06	1,270,807	4	http://sourceforge.net/projects/poptray/
24	Alfresco Content Management	07/08/05	03/09/09	1,258,751	19	http://sourceforge.net/projects/alfresco/
25	e107	09/30/02	12/09/08	1,206,188	75	http://sourceforge.net/projects/e107/
26	Glade2/Gtk+	01/05/04	05/02/08	1,173,856	2	http://sourceforge.net/projects/gladewin32/
27	HSQldb	03/21/01	06/02/08	1,155,736	35	http://sourceforge.net/projects/hsqldb/
28	WebCalendar	03/22/00	09/27/08	1,010,534	7	http://sourceforge.net/projects/webcalendar/
29	Lazarus	09/03/03	10/20/08	1,400,48	5	http://sourceforge.net/projects/lazarus/
30	MozillaPL.org	07/17/01	11/30/07	985,676	8	http://sourceforge.net/projects/mozilla-pl/
31	Openbravo ERP	03/09/06	03/20/09	954,287	82	http://sourceforge.net/projects/openbravo/
32	Plone	02/21/02	02/16/08	947,162	92	http://sourceforge.net/projects/plone/
33	XDoclet	07/18/01	06/11/05	936,743	34	http://sourceforge.net/projects/xdoclet/

34	dotProject	02/28/01	07/29/08	928,220	8	http://sourceforge.net/projects/dotproject/
35	Snitz Forums 2000	07/26/00	03/17/09	899,432	12	http://sourceforge.net/projects/sf2k/
36	ZeosLib	09/19/01	11/01/08	829,792	15	http://sourceforge.net/projects/zeoslib/
37	Nhibernate	02/11/03	03/16/09	776,875	24	http://sourceforge.net/projects/nhibernate/
38	phpPgAdmin	10/05/01	12/18/08	745,580	8	http://sourceforge.net/projects/phpPgAdmin/
39	Mambo	04/19/01	04/14/08	740,500	1	http://sourceforge.net/projects/mambo/
40	Jadclipse	11/16/01	04/11/07	740,198	3	http://sourceforge.net/projects/jadclipse/
41	TikiWiki CMS/Groupware	10/07/02	10/17/08	720,774	413	http://sourceforge.net/projects/tikiwiki/
42	PNotes	07/19/03	01/04/09	719,950	9	http://sourceforge.net/projects/turbocash/
43	TurboCASH Accounting	07/19/03	01/04/09	708,478	9	http://sourceforge.net/projects/turbocash/
44	TOra	12/15/00	12/15/00	690,926	15	http://sourceforge.net/projects/tora/
45	phplist	09/26/03	01/28/09	627,832	7	http://sourceforge.net/projects/phplist/
46	FreeBASIC Compiler	10/22/04	08/11/08	609,483	8	http://sourceforge.net/projects/fbc/
47	Dev-PHP	05/31/02	01/04/09	598,227	5	http://sourceforge.net/projects/devphp/
48	RoundCube Webmail	05/19/05	03/10/09	564,975	6	http://sourceforge.net/projects/roundcubemail/
49	Yet another Bulletin Board	07/30/00	01/04/09	534,072	25	http://sourceforge.net/projects/yabb/
50	SWIG	01/19/00	01/31/09	509,491	36	http://sourceforge.net/projects/swig/
51	MacOS X HTTP Mail Plugin	02/22/03	08/10/06	499,035	1	http://sourceforge.net/projects/httpmail-plugin/
52	Magic Mail Monitor 3	12/12/02	01/28/09	495,614	3	http://sourceforge.net/projects/mmm3/
53	Ariane RPG	01/04/00	02/22/09	446,552	23	http://sourceforge.net/projects/arianne/
54	TUTOS	07/10/00	01/05/09	441,182	5	http://sourceforge.net/projects/tutos/
55	opentaps open source ERP+CRM	08/10/05	03/11/09	440,162	37	http://sourceforge.net/projects/opentaps/
56	Grisbi	10/31/03	01/26/09	433,363	35	http://sourceforge.net/projects/grisbi/
57	phpGroupWare	06/23/00	08/14/07	422,995	23	http://sourceforge.net/projects/phpgroupware/
58	Rapid Miner	07/10/04	11/23/08	408,333	19	http://sourceforge.net/projects/yale/
59	Crystal Space 3D SDK	12/09/99	05/28/08	403,801	34	http://sourceforge.net/projects/crystal/
60	ADempiere ERP Business Suite	09/09/06	04/10/09	388,340	86	http://sourceforge.net/projects/adempiere/
61	The Progect Project Manager	08/04/00	09/01/04	375,999	8	http://sourceforge.net/projects/progect/
62	SugarCRM	04/23/04	01/28/09	361,497	26	http://sourceforge.net/projects/sugarcrm/
63	RemoteCalendars	07/20/05	05/19/07	350,563	22	http://sourceforge.net/projects/remotecalendars/
64	PhpWiki	05/26/00	03/17/08	342,563	17	http://sourceforge.net/projects/phpwiki/
65	JWebMail / Java Webmail	01/31/00	10/31/08	338,872	3	http://sourceforge.net/projects/jwebmail/
66	SW Test Automation Framework	08/08/01	12/10/08	328,554	6	http://sourceforge.net/projects/staf/
67]project-open[07/23/03	01/31/09	307,780	16	http://sourceforge.net/projects/project-open/
68	Simple PHP Blog	04/09/04	09/23/07	293,111	4	http://sourceforge.net/projects/sphpblog/
69	xajax PHP and Javascript library	05/24/05	08/06/08	287,578	6	http://sourceforge.net/projects/xajax/
70	phpCollab	02/12/02	07/13/08	281,832	8	http://sourceforge.net/projects/phpcollab/

71	Attachment Mod	11/03/02	09/06/06	275,672	1	http://sourceforge.net/projects/acydmods/
72	Market Analysis System	06/12/00	07/29/04	270,325	2	http://sourceforge.net/projects/eiffel-mas/
73	Ajax JSP Tag Library	06/02/05	03/22/09	266,970	11	http://sourceforge.net/projects/ajaxtags/
74	Claws Mail	04/18/01	03/06/09	264,471	10	http://sourceforge.net/projects/sylpheed-claws/
75	WebGUI	04/16/02	03/26/09	255,484	3	http://sourceforge.net/projects/pbwebgui/
76	Yet Another Forum.NET	09/18/03	04/01/08	255,368	14	http://sourceforge.net/projects/yafdotnet/
77	Castle Project	11/18/04	03/10/09	243,479	9	http://sourceforge.net/projects/castleproject/
78	Task Coach	02/07/05	03/13/09	243,352	2	http://sourceforge.net/projects/taskcoach/
79	NOCC	09/30/00	01/01/09	241,194	11	http://sourceforge.net/projects/noccl/
80	openCRX - Enterprise Class CRM	11/18/03	03/26/09	240,645	2	http://sourceforge.net/projects/opencrx/
81	webERP web-based ERP Accounting	07/01/03	03/30/09	234,172	9	http://sourceforge.net/projects/web-erp/
82	Openbravo POS	01/04/05	03/04/09	228,027	10	http://sourceforge.net/projects/openbravopos/
83	GFP	14/04/04	10/15/07	226,008	11	http://sourceforge.net/projects/gfd/
84	BugTracker.NET	11/10/02	03/16/09	212,407	1	http://sourceforge.net/projects/bitnet/
85	SAGA GIS	02/20/04	07/01/08	211,847	12	http://sourceforge.net/projects/saga-gis/
86	Cybera - Cyber cafe administration	10/07/04	12/10/06	209,154	2	http://sourceforge.net/projects/cybera/
87	ainbowPortal	11/10/02	11/23/06	194,026	4	http://sourceforge.net/projects/rainbowportal/
88	KMyMoney	04/16/00	09/13/08	188,937	10	http://sourceforge.net/projects/kmymoney2/
89	Turquaz Financial Accounting	05/01/03	02/20/08	186,232	14	http://sourceforge.net/projects/turquaz/
90	iGnash	05/14/01	02/27/09	176,102	6	http://sourceforge.net/projects/ignash/
91	Mahogany mail and news client	02/28/00	08/06/06	175,493	8	http://sourceforge.net/projects/mahogany/
92	MonoCalendar	02/25/05	01/31/07	173,003	2	http://sourceforge.net/projects/monocalendar/
93	Columba	02/22/01	04/21/07	168,147	16	http://sourceforge.net/projects/columba/
94	mvnForum	10/24/02	11/19/08	167,534	29	http://sourceforge.net/projects/mvnforum/
95	Help Center Live	10/31/03	07/03/08	162,225	2	http://sourceforge.net/projects/helpcenterlive/
96	OneOrZero Helpdesk	11/09/04	03/01/09	161,468	5	http://sourceforge.net/projects/oneorzero/
97	Fully Modded phpBB	02/06/03	01/11/09	150,710	1	http://sourceforge.net/projects/phpbbfm/
98	Yuza Open Erp	07/07/05	03/07/09	150,252	1	http://sourceforge.net/projects/yuza/
99	Slash	04/07/00	08/18/06	141,352	13	http://sourceforge.net/projects/slashcode/
100	PNphpBB2	03/21/03	06/10/06	137,261	8	http://sourceforge.net/projects/pnphpbb2/
101	phPay	11/04/01	04/14/03	131,372	1	http://sourceforge.net/projects/phpay/
102	EclipseTrader	08/09/04	05/18/07	127,817	2	http://sourceforge.net/projects/eclipse trader/
103	Hebrew Mozilla	12/20/01	07/30/07	127,486	4	http://sourceforge.net/projects/hebmoz/
104	Phoenix Mail	05/31/00	12/07/05	123,238	8	http://sourceforge.net/projects/phxmail/
105	SQL-Ledger	04/04/00	03/16/07	119,156	1	http://sourceforge.net/projects/sql-ledger/
106	TestLink	09/25/03	03/23/09	115,417	11	http://sourceforge.net/projects/testlink/
107	Spamato Spam Filter System	04/08/05	08/01/07	114,208	6	http://sourceforge.net/projects/spamato/

108	Information Resource Manager	11/11/00	10/23/07	112,419	4	http://sourceforge.net/projects/irm/
109	PHP Point Of Sale	02/14/03	05/07/06	111,298	1	http://sourceforge.net/projects/phppointofsale/
110	Track + Issue Tracker	11/05/01	02/21/08	108,976	8	http://sourceforge.net/projects/trackplus/
111	QuickFIX	10/11/01	11/08/06	107,754	9	http://sourceforge.net/projects/quickfix/
112	hipergate CRM	09/10/03	01/10/09	106,052	6	http://sourceforge.net/projects/hipergate/
113	BORG Calendar	10/26/03	03/14/09	102,065	2	http://sourceforge.net/projects/borg-calendar/
114	Lx-Office	08/05/04	12/19/08	98,973	4	http://sourceforge.net/projects/lx-office/
115	TWiki Enterprise Wiki	03/16/00	12/19/08	96,840	16	http://sourceforge.net/projects/twiki/
116	TA-Lib: Technical Analysis Library	07/28/00	09/20/07	88,739	17	http://sourceforge.net/projects/ta-lib/
117	phpScheduleIt	11/21/03	02/20/09	84,845	8	http://sourceforge.net/projects/phpscheduleit/
118	FreeMED	01/14/00	05/21/07	82,766	14	http://sourceforge.net/projects/freemed/
119	Barbecue - Java barcode generator	04/04/03	05/06/07	80,404	5	http://sourceforge.net/projects/barbecue/
120	OpenBiblio	03/29/02	05/09/08	74,458	9	http://sourceforge.net/projects/obiblio/
121	vym - view your mind	01/03/05	10/29/08	69,686	4	http://sourceforge.net/projects/vym/
122	zenTrack - project/bug tracking software	03/14/01	01/26/09	64,556	9	http://sourceforge.net/projects/zentrack/
123	OSSUITE	10/18/02	07/07/03	60,510	5	http://sourceforge.net/projects/ossuite/
124	JGraphT	07/24/03	09/29/08	57,949	16	http://sourceforge.net/projects/jgraph/
125	Evaristo	09/16/03	09/05/08	56,715	2	http://sourceforge.net/projects/evaristo/
126	WebCollab	03/09/03	02/07/09	55,796	1	http://sourceforge.net/projects/webcollab/
127	bitweaver	06/13/05	01/26/09	55,090	81	http://sourceforge.net/projects/bitweaver/
128	Tiny ERP	03/25/05	02/12/08	54,237	1	http://sourceforge.net/projects/tinyerp/
129	OpenRPT report writer by xTuple	03/03/05	01/15/09	52,332	8	http://sourceforge.net/projects/openrpt/
130	PHP Newswriter 2005	03/31/04	02/01/07	49,895	2	http://sourceforge.net/projects/newswriter2005/
131	Pebble	04/19/03	02/10/09	47,731	2	http://sourceforge.net/projects/pebble/
132	Very Quick Wiki	08/21/01	03/01/09	46,631	16	http://sourceforge.net/projects/veryquickwiki/
133	CK-ERP	11/22/03	01/31/09	41,789	1	http://sourceforge.net/projects/ck-erp/
134	Merchant of Venice	05/15/02	02/25/07	38,935	9	http://sourceforge.net/projects/mov/
135	MyPhpMoney	02/07/02	06/06/07	38,087	9	http://sourceforge.net/projects/myphpmoney/
136	Covide	12/24/03	02/02/09	37,975	15	http://sourceforge.net/projects/covide/
137	iTrade - Trading and Charting System	01/08/05	01/03/08	36,473	8	http://sourceforge.net/projects/itrade/
138	Neoglia	01/13/05	03/27/09	36,227	34	http://sourceforge.net/projects/neoglia/
139	uEngine BPM	12/18/03	01/07/09	35,671	17	http://sourceforge.net/projects/uengine/
140	Openl	07/01/05	08/15/08	34,908	11	http://sourceforge.net/projects/openl/
141	Ohioedge CRM + BPM Server	12/19/02	02/25/07	31,471	2	http://sourceforge.net/projects/ohioedge/
142	Plazma	07/01/05	02/25/09	30,237	1	http://sourceforge.net/projects/plazma/
143	Finance:Quote	04/02/00	10/26/08	29,409	12	http://sourceforge.net/projects/finance-quote/
144	Wikepage/Blog Hybrid Engine	01/17/05	10/14/08	27,833	3	http://sourceforge.net/projects/wikepage/

145	openPHNuke	08/31/01	12/06/08	26,544	6	http://sourceforge.net/projects/dev-openphpnuke/
146	PhPepperShop	09/12/01	12/20/07	25,996	3	http://sourceforge.net/projects/phpeppershop/
147	CatMDEdit-metadata editor	09/28/04	04/08/08	21,511	1	http://sourceforge.net/projects/catmdeedit/
148	osCSS	04/27/05	09/01/08	15,412	8	http://sourceforge.net/projects/oscss/
149	WORK system CMS e-commerce	02/06/04	02/09/09	11,413	1	http://sourceforge.net/projects/worksystem/
150	Krang	02/24/04	08/25/08	11,158	7	http://sourceforge.net/projects/krang/

Appendix C

Respondent's Responses Sample

Respondent's Responses Sample

No	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	2010-03-28 08:49:49	5+ years	Master's degree	Active Developer	Part-time	Plone	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	2010-05-03 01:43:21	5+ years	Bachelor's degree	Core Developer	Part-time	Worldforge	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
9	2010-05-11 09:25:43	3- 5 years	Bachelor's degree	Core Developer	Part-time	ADempiere	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
8	2010-04-05 02:48:14	5+ years	Bachelor's degree	Other	Other	I localize for over 20 projects!	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	No
19	2010-05-17 19:13:27	1-3 years	Other	Passive Developer	Part-time	LimeSurvey	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No	No	Yes	Yes
5	2010-05-02 10:49:06	3- 5 years	Master's degree	Project Manager.	Part-time	Sabily	Yes	Yes.	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
6	2010-03-28 18:27:39	< 1 year	Bachelor's degree	Other	Other		Yes	No	No	No	No	No	Yes	Yes	Yes	No	No	Yes
7	2010-05-01 11:00:53	Less than 1 year	Other	Core Developer	Dedicated full-time	alfanous	No	No	No	Yes	Yes	No	No	No	Yes	No	No	Yes
10	2010-03-28 14:19:28	5+ years		Project leader	Part-time	Plone	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
11	2010-03-31 02:38:46	5+ years	Master's degree	Project leader	Dedicated full-time]project-open[Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes

12	2010-03-30 16:06:06	5+ years	Master's degree	Project leader	Other	PHP_Beautifier and ruby-statsample	Yes	No	Yes	No	No	No	No	Yes
13	2010-04-05 07:57:53	5+ years	Other	Core Developer	Part-time	scalacs, maven- scala-plugin, ... see http://github.com/davidB	Yes	Yes	Yes	No	Yes	No	No	Yes
15	2010-03-25 12:32:56	5+ years	Bachelor's degree	Project leader	Dedicated full-time	Mozilla	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
16	2010-03-28 15:17:34	5+ years	Other	Core Developer	Part-time		Yes	Yes	No	Yes	No	Yes	PM's	Yes
17	2010-03-28 14:12:22	5+ years	Other	Project leader	Part-time	Plone, Zope2, Zope Toolkit, Distribute, Chameleon, ... many more	Yes	Yes	Yes	No	Yes	Yes	Blog, Planets, Twitter, Real- life sprints, conferences	Yes
18	2010-03-28 15:19:09	5+ years	Bachelor's degree	Core Developer	Dedicated full-time	Tikiwiki, other smaller or yet to be made public projects as well	Yes	Yes	Yes	Yes	No	Yes	No	Yes
20	2010-05-03 07:09:49	5+ years	Bachelor's degree	Active Developer	Other	ImageFuser, KImageFuser, Hugin, Avidemux, Panini	Yes	Yes	No	No	Yes	Yes	No	No

21	2010-03-22 02:44:02	5+ years	Master's degree	Core Developer	Part-time	SquirrelMail	Yes	Yes	Yes	Yes	No	Yes	No	Yes
22	2010-03-22 02:55:29	1-3 years		Active Developer	Part-time	WebCalendar	Yes	Yes	No	Yes	Yes	No	No	Yes
23	2010-03-22 04:01:31	1-3 years	Master's degree	Passive Developer	Part-time		Yes	Yes	No	Yes	No	Yes	No	Yes
24	2010-03-22 09:32:02	5+ years	Bachelor's degree	Project leader	Part-time	MinGW (also participate in GNU- Troff)	Yes	No	No	No	No	Yes	No	No
25	2010-03-23 06:12:05	5+ years	Doctoral degree	Project leader	Dedicated full-time	VASSAL	Yes	Yes	No	Yes	No	Yes	flat, unthread forums	Yes
26	2010-03-24 08:43:56	3- 5 years	Bachelor's degree	Active Developer	Part-time	Zentrack	Yes	Yes	Yes	Yes	Yes	No	No	Yes
27	2010-03-25 02:55:00	5+ years	Master's degree	Project leader	Part-time	Gammu	Yes	Yes	No	Yes	No	Yes	No	Yes
28	2010-03-25 06:17:48	5+ years	Master's degree	Project leader	Part-time	ComingNext	No	No	No	No	Yes	No	No	Yes

44	44	45	46	47	47	48	51	52	53	54	55	56	56	56	
Yes	Yes	No	No	Yes	No	Yes	release manager	minor each month, major each	Release when its ready	No	Yes	No	Subversion	command line subversion	unittest (python library)
Yes	No	No	Yes	No	Yes	Core Developer	Core Developer	Every quarter	Release often and early	Yes	Yes	git	git		
Yes	No	No	Yes	No	Yes	Project Leaders	Project Leaders			No	No	Subversion	Subclipse		
No	No	Yes	Yes	No	N/A	Project Leaders	Varies a lot between projects	Release when its ready	Release when its ready	Yes	No	Mostly SVN, some have moved to git or bzt, only have a few still using	svnX, GitX, repo web views, but mostly CLI	gettext process, Pootle	Pootle, Damned Lies
No	No	No	No	No	No	Project Leaders	Every quarter	Release when its ready	Release when its ready	Yes	No	Subversion	SVN command line	none	none
Yes	No	No	No	No	No	Project Leaders	Every six months			No	No	Bazaar version	bazaar client	Many	Many
Yes	Yes	No	No	Yes	No	Core Developer		Fixed features	Fixed features	No	No	CVS	Command Line CVS		
No	No	N/A	Yes	No	No	Core Developer		Release when its ready	Release when its ready	No	No	Subversion	Svn workbench		
Yes	Yes	Yes	Yes	Yes	Yes	Dedicated Manager	Every six months	Fixed features	Fixed features	No	Yes	Subversion	command line. There was no option for "none of the above"		
Yes	No	Yes	Yes	Yes	No	Project Leaders	Every year	Release when its ready	Release when its ready	No	No	CVS			

No	No	Yes	Yes	No	No	Project Leaders	Continue	Release often and early	No	Yes	No	No	SVN and GIT	console tools	Unit: Test on ruby		
Yes	No	Yes	No	No	No	Dedicated Manager	on feature, bug-fix addition	Release when its ready	Yes	Yes	No	No	git	git command line	testing, specs, scalacheck, maven-invoker-plugin		
No	No	Yes	Yes	No	Yes	Project Leaders	Every month	Release often and early	No	Yes	Yes	Yes	mercurial	hgweb	litmus	bugzilla	
Yes	Yes	No	Yes	No	No	Core Developer	When there is something worth	Release when its ready	Yes	Yes	No	No	Subversion				
Yes	Yes	Yes	Yes	Yes	Yes	Dedicated Manager	Every month	Fixed monthly date for maintenance releases, "release when it's ready" for new feature releases	No	No	Yes	Yes	Subversion	Textmate SVN bundle	Python unittest, zope.testing, various test case products, testbrowser, QUnit, Selenium, buildbot, Hudson	diff mails to mailing lists	
Yes	No	No	No	Yes	No	Community Developer	Every six months	Fixed dates	No	Yes	Yes	Yes	Subversion	up to contributors	PHPUnit	Mailing lists with subversion commits	
Yes	Yes	No	Yes	No	No	Project Leaders	depends on the project	Release often and early	No	Yes	No	No	Subversion	svn, bzt, hg			

No	No	No	No	No	No	Core Developer	Every quarter	Release when its ready	No	No	Yes	No	Subversion	Subversion command line		
Yes	No	N/A	N/A	Yes	No	Project Leaders		Release when its ready	No	No	No	No	CVS	Cygwin	web browsers	
No	No	Yes	Yes	Yes	No	Project Leaders		Release when its ready	Yes	No	No	No	Subversion			
No	No	N/A	N/A	No	N/A	Elected Candidate		Release when its ready	No	No	No	No	CVS	command line via ssh		
Yes	No	Yes	Yes	Yes	No	Core Developer	Every month	Release when its ready	No	No	Yes	No	Subversion	command line SVN client	JUnit	
Yes	Yes	No	No	Yes	Yes	Dedicated Manager	Every year	Fixed features	No	No	Yes	Yes	Subversion	Subclipse		Smart Bear Code Collaborator
Yes	Yes	Yes	No	Yes	Yes	Project Leaders	Every month	Release when its ready	No	No	Yes	No	Git	Git	CTest	
No	No	No	No	No	No	Project Leaders	Every month	Fixed features	No	No	No	No	git	git		

	64	64	Comments	First name	Email address
Buildout			A short description of used terminology (in some case we use different terms).	Riccardo Lemmi	rlemmi@users.sourceforge.net
automake			Allow multiple selections for source control.	Erik Hjortsberg	erik.hjortsberg@gmail.com
RSS	Eclipse	XCode	It's very objective. I hope it would increase what you are expected It doesn't really consider localizers, and we're a key part of any dev project. Also, you didn't consider that people might contribute to more than one FLOSS project. I'm not sure if my answers are even useful statistically, since the questions seem aimed exclusively at people producing source code, not localizing it.	Juan Carlos Perez	jc-perez@users.sourceforge.net
don't know	none	none	This has been a very thought provoking survey. I will talk to the other developers about getting a standard testing regime for LimeSurvey	Eric T Cruiser	eric_t_cruiser@users.sourceforge.net
Many	--	Many	Good	Abdelmonam Kouka	abdelmonam.kouka@gmail.com
python setuptools	Eclipse	eclipse-pydev	How will we know the results about the survey? I suggest to add more infos about used programming language	**Mya** Assem Chelli	marylly@users.sourceforge.net assem.ch@gmail.com
HUDSON, buildbot			need more explanations for questions for those that don't know all the tools or techniques. Also need more "none of the above" options.	Geir B	elvix@users.sourceforge.net
-	Integrated in OpenACS	Emacs...	Our product is an ERP system with more then 1.000.000 lines of code (probably the biggest open-source Web-application in the world). A full test (documented test cases) of the entire application takes 3-4 days. The best we've found: "continuous" testing with rigid control of deltas, and release the code after large rollouts (>50 users) - the code will surely work then :) Frank	Frank Bergmann	fraber@users.sourceforge.net

rake, hoe and jeweler, pear	rdoc and php_documentat or		Provides multiple options for CVS. You should add console tools as options.	Claudio Bustos	cdx@users.sourceforge.net
maven, sbt, hudson	javadoc+apiviz, vscaaldaoc	eclipse, idea	Some complementary questions you could add : * do you use a project hosting : sourceforge, github, googlecode, assembla, codehaus. apache.... * do you continuous integration server : hudso, teamcity,... * do you work on : standalone project, plugin, library you SCM only list centralized SCM, and not DVCS (increase in popularity) like git, mercurial,... Good luck	David Bernard	dwayneb@users.sourceforge.net
buildbot, tinderbox	?	eclipse, quanta, textmate, notepad++	sometimes it was hard to match answers with reality. For degree you started with bachelor - a lot of our community members are dropouts, you missed them. For RCS you focused on CVS/SVN - I think few FLOSS projects still use those old ones. We use tons of tools, and we release many products, so I focused on Firefox, but in reality I work on many projects/products inside Mozilla.	Zbigniew Braniecki	e-gandaif@users.sourceforge.net
		NotePad++	The questions seemed to assume you only work on a single OSS project.	Richard	whoisrich@users.sourceforge.net
Python, buildout, distribute/etuptools	Sphinx	lots, personally Textmate	The survey assumes one is only active in one project. I happen to be active in more than a dozen of them, which all have different approaches. My role in those projects is also different, from founder, release manager, core developer to projects I contributed to in the past. The answers in this survey are for the Plone project.	Hanno Schlichting	hannosch@users.sourceforge.net
		up to contributors, most use V1	There should be a clearer distinction between automated tests and manual tests.	Louis-Philippe Huberdeau	lphuberdeau@users.sourceforge.net
XCode and cmake		XCode	This survey focusses on one project. I'm "owner" of 2 projects and I am "core member" of 3 other projects. It's a bit difficult to give precise answers to some questions.	Harry van der Wolf	hvdwolf@users.berlios.de

					Thijs Kinkhorst	kink@users.sourceforge.net
					Bruce	bbannon@users.sourceforge.net
					Maciej Drwal	maciekd@users.sourceforge.net
					MinGW Project Shell Managment	mingw@users.sourceforge.net
Make	Javadoc	Eclipse			Joel Uckelman	uckelman@users.sourceforge.net
		Eclipse			Matt Clark	donkeybandit@users.sourceforge.net et
CMake	Doxygen + Sphinx	gvim			Michal Čihař	nijel@users.sourceforge.net
perl		notepad++			Michael Prager	michaelprager@users.sourceforge.net et

Appendix D
Publication



Electronic Communications of the EASST

is a peer-reviewed, scientific and open access journal

ISSN 1863-2122

HOME ABOUT LOG IN REGISTER SEARCH VOLUMES

OPEN JOURNAL SYSTEMS

Journal Help

USER

Username

Password

Remember me

JOURNAL CONTENT

Search

All

Browse

- [By Volume](#)
- [By Author](#)
- [By Title](#)

INFORMATION

- [For Readers](#)
- [For Authors](#)
- [For Librarians](#)

Home > Volumes > **Volume 33: Foundations and Techniques for Open Source Software Certification 2010**

Volume 33: Foundations and Techniques for Open Source Software Certification 2010

Table of Contents

Preface

Preface

[PDF](#)

Luis Soares Barbosa, Antonio Cerone, Siraj Ahmed Shaikh

Articles

Component Certification as a Prerequisite for Widespread OSS Reuse

[ABSTRACT PDF](#)

George Kakarontzas, Panagiotis Katsaros, Ioannis Stamelos

Damages and Benefits of Certification: A perspective from an Independent Assessment Body

[ABSTRACT PDF](#)

Mario Fusani, Eda Marchetti

Security in Open Model Software with Hardware Virtualisation – The Railway Control System Perspective

[ABSTRACT PDF](#)

Johannes Feuser, Jan Peleska

Security Evaluation and Hardening of Free and Open Source Software (FOSS)

[ABSTRACT PDF](#)

Robert Charpentier, Mourad Debbabi, Dima Alhadidi, Azzam Mourad, Nadia Belblidia, Amine Boukhtouta, Aiman Hanna, Rachid Hadjidj, Hakim Kaitouni, Marc-André Laverdière, Hai Zhou Ling, Syrène Tili, Xiaochun Yang, Zhenrong Yang

Methodologies and Tools for OSS: Current State of the Practice

[ABSTRACT PDF](#)

Zulqarnain Hashmi, Siraj Ahmed Shaikh, Naveed Ikram

Integrating Data from Multiple Repositories to Analyze Patterns of Contribution in FOSS Projects

[ABSTRACT PDF](#)

Sulayman K Sowe, Antonio Cerone

Clang and CocciNelle: Synergising program analysis tools for CERT C Secure Coding Standard certification

[ABSTRACT PDF](#)

Mads Chr. Olesen, Rene Rydhof Hansen, Julia L. Lawall, Nicolas Palix

Open Source Verification under a Cloud

[ABSTRACT PDF](#)

Peter T Breuer, Simon Pickin

Using Free/Libre Open Source Software Projects as E-learning Tools

[ABSTRACT PDF](#)

Antonio K Cerone, Sulayman K Sowe

Testing as a Certification Approach

[ABSTRACT PDF](#)

Alberto Simões, Nuno Carvalho, José João Almeida

GUI Inspection from Source Code Analysis

[ABSTRACT PDF](#)

Joao Carlos Silva, José Creissac, Joao Saralva

Safe Integration of Annotated Components in Open Source Projects

[ABSTRACT PDF](#)

Sergio Areias, Daniela Da Cruz, Pedro Rangel Henriques, Jorge Sousa Pinto

A Deductive Verification Platform for Cryptographic Software

[ABSTRACT PDF](#)

Manuel Barbosa, J. Pinto, J.-C. Filliatre, B. Vieira

