Electronic Communications of the EASST
Volume 33 (2010)

Proceedings of the Fourth International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2010)

Methodologies and Tools for OSS: Current State of the Practice

Zulqarnain Hashmi, Siraj A. Shaikh and Naveed Ikram

11 pages

Guest Editors: Luis S. Barbosa, Antonio Cerone, Siraj A. Shaikh
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Methodologies and Tools for OSS: Current State of the Practice

Zulqarnain Hashmi[1], Siraj A. Shaikh[2] and Naveed Ikram[1]

[1] zulqarnain@iiu.edu.pk, naveed.ikram@iiu.edu.pk
Department of Software Engineering,
Faculty of Basic and Applied Sciences,
International Islamic University, Islamabad, Pakistan

[2] s.shaikh@coventry.ac.uk
Department of Computing and the Digital Environment,
Faculty of Engineering and Computing,
Coventry University, Coventry, United Kingdom

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 paper presents the results of a survey of the OSS developer community with a view to gain insight of peer review, testing and release management practices, along with the current tool sets used for testing, debugging and, 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 survey will initiate a useful discussion and allow the community to identify further process improvement opportunities for producing better quality software.

Keywords: Open Source, Testing, Debugging, Release Management, Peer Review.

1 Introduction

Open Source Software (OSS) is becoming popular both in business communities and academic sectors. The OSS movement has proved its worth with notable products such as Linux, Apache, MySQL and Mozilla, to name a few.

OSS development is typically initiated by a small group of people [SFF+06] and can be distinguished from traditional development in terms of volunteers involved in the development of software dictated by their need and interest, as opposed to a dedicated team of paid developers guided by some (usually profit-making) commercial product. Such volunteers choose when and what they want to work on with typically a very loose heirarchy, as opposed to their paid counterparts. The expectation of most OSS projects is that there is less support in terms of development tools, no or very less formal design, improper project development planning, a fixed list of deliverables is not available and finally no structured testing or quality assurance of the final product. OSS projects are also less likely to be supported by project management, metrics, estimation and scheduling tools as there is no need for strict deadlines and balancing budgets [RFL05, Rob02].

OSS projects have been criticised for lack of clear and open detail of development processes. Studies on Apache and Mozilla [MFH99, MFH02, RM02] usually give an informal description

of development processes which cannot be usefully replicated.

Our observation discovers that sources of information on the community, project history, work roles and task prescription provided on several OSS project websites appear mind-numbing and ambiguous. A need for standard and clear development practices has been acknowledged [Sca03]. Sharing clear and open description of development processes, with a view to further improvisation and reuse is certainly of great interest to the wider OSS community [Mic05, RM02]. Our effort is aimed at better understanding some of the development processes and behaviour within a set of OSS projects.

We present a 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 [Sou09] and Launchpad [Lau09] 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; it is essentially an objective measure independent of our influence [Mic05].

The hope is that this work will allow the wider community to identify process improvement for better quality and critical software. The importance of quality in OSS due to development practices has already been acknowledged [SC09]. Such an insight is important to appreciate the obstacles to overcome to introduce certification and more rigour into the development and testing processes. Moreover, attempts to introduce the use of formal modelling and verification within OSS developmen practices has also been suggested [CS08], though challenges have also been identified as to who and where to initiate such changes within the OSS community; addressing such challenges is of interest to us in this paper and is essentially the next step.

1.1 Rest of this paper

The rest of this paper is organised as follows. Section 2 describes some of the related work. Some past observations and the relevant trends observed are brought to attention. Section 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. Section 4 presents the results of the survey. Some trends of interest are highlighted though the majority of the results serve to affirm traditional perceptions of the OSS community. Section 5 concludes the paper and promises some future work.

2 Related work

Organisational structures, technical roles and career opportunities within the OSS community have been widely studied [Sca07, YK]. Traditionally software engineers have been restricted to roles like requirements analyst, software designer, programmer or code tester. 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. Some of the recognised 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.

Several developers and users examining the source code is one of the fundamental principles that underlies OSS. Some reports show that peer review on some OSS has been performed by millions of developers [GBBZ03]. Such code reviews are mostly done before and after any source code is committed to the repository [HS02], performed in a distributed, asynchronous manner. They are certainly more extensively acknowledged and accepted as part of the organisational 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 recognised in software engineering generally for its crucial role [Ema01, HS02].

In a survey [Sta02], about 9% of OSS developers claimed the peer review of the entire source code, whereas peer review of most of the code was pointed out by 50% of the developers. 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 reviewers 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 [DRW03].

Testing is an essential part of the software development life cycle. Recent studies establish the uniqueness of the OSS development model with exceptionally high user involvement and structured approach for flaw/bug handling process, in the context of testing for OSS [OMK08]. Unit testing is the most frequent in OSS development. 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 [HS02, GA04]. With confidence in code peer reviewed, many OSS developers are content with only minor testing [Sta02]. Some other sources go as far as to claim that over 80% of OSS developers dont have any plans of testing [ZE00].

There is no specific evidence with regards to automated tools but debuggers are widely acknowledged [ZE00]. For regression tests, about 48% of OSS projects follow baseline testing, whereas proportion is relatively higher in mega projects [ZE03]. A study conducted on the Apache project revealed that no system testing or regression was performed [MFH02]. Further analysis reports that while regression test suites were available for Apache they were not actually mandatory [Ere03]. This complements with suggestions that improvement is required in quality assurance practices, applied processes and project success criteria [OMK08].

Release management is a vital part in OSS development. Michlmayar [Mic07] presents a comparative study on release management to find that it can be categorised 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 stabilised 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 stabilised release [MHP07].

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.

3 Research Design

The research method used in this study is essentially a survey which is most common for generating primary data. This survey is descriptive in nature and lies in cross-sectional time dimensional category. The unit of analysis is essentially individuals and OSS developers are the respondents for this survey. The main objectives of the survey are to determine the development processes and developmental tools being used in OSS projects.

Our population is open source developers and targeted populations are developers of recognised OSS initiatives. The most important source to collect information about the development processes and tools used in OSS projects are 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 [Sou09] and Launchpad [Lau09]. We have chosen a systematic sampling method where 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 a measure as it is objective and dependent on the users [Mic05].

We designed an online questionnaire consisting of 33 questions in total. We drew inspiration from [KENU07] for questions relating to peer review and testing of software. Validity and reliability are the main priorities in surveys. There is a need of for pilot testing to assess the questionnaire clarity, understandability, comprehensiveness and acceptability. Surveys should be adequately pre-tested to check that the respondents understand the meaning of the questions or statements and to gauge whether test items are at an appropriate level of difficulty.

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. Several changes were made to improve the experience as per the feedback.

A detailed search was undertaken to identify projects which existed with the same name under different domains. 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). 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.

4 Results and Analysis

This section presents the results 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

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. 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. 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 and community manager roles. 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 *occassional*.

When asked about information provision and dissemination for their OSS projects, over 96% of the respondents claimed that their project has a dedicated website. Of other similar resources, 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.

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%).

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.

4.2 Peer Review

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.

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.

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. Ony 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.

4.3 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. Over 42% of the projects are said to have some formal testing procedure.

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. 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*. 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%.

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

4.4 Release management

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.

Of the projects surveyed, just under 30% release every six months, with 11% releasing every quarter and a similar percentage every year. 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*.

The decision to release is as important as the frequency: 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.

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 as in providing no central source base and different branches holding different parts of the code.

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.

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.

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 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. 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.

4.5.7 Integrated Development Environment

Eclipse has been recognised as the most common development platform amongst the community surveyed. Other notable tools mentioned include CodeLite, VisualStudio, ReSharper, Quanta HTML editor, TextMate, Kate, Delphi, Lazarus, Komodo IDE, Notepad++, Qt Creator, Vim, Emacs, Xcode, NetBeans, Eclipse-Pydev and PyPaPi.

4.6 Analysis

With over half the respondents having over 5 years of experience with OSS, 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 with 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 a 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 an wide range of communication tools and strategies with nearly all having a dedicated website. Feature lists, mailing lists, user and developer documentation are some of the 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 survey results offers 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 fall in this category.

5 Conclusion

The work presented in this paper came out of a desire to understand the OSS developer community better and the state of current development practices. The accuracy of the survey results presented in this paper is undoubtedly subject to the survey design and the target population. It serves however to provide a snapshot which is both useful and indicative of further inquiry.

The motivation behind this work follows from earlier work [CS08] 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 results of this paper 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.

5.1 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. Following the survey, we are currently in the process of setting up a shorter follow-up survey to explore the perceptions of formal methods and more rigorous methods alike for adoption by the community. Aspects of software modelling and verification, assurance and certification will be explored. We hope to report on the results of this follow-up survey soon. These results will undoubtedly provide us with a platform for more concrete proposals for change.

Acknowledgements: The authors would like to thank Shahida Bibi at International Islamic University for her assistance with data collection for this paper.

Bibliography

[CS08] 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.

[DRW03] 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.

[Ema01] K. E. Emam. Software Inspection Best Practices. *Agile Project Management Advisory Service* 2(9), 2001.

[Ere03] J. R. Erenkrantz. Release management within open source projects. *Proceedings of the Third Workshop on Open Source Software Engineering, Portland, Oregon*, 2003.

[GA04] C. Gacek, B. Arief. The Many Meanings of Open Source. *IEEE Software* 21(1):34–40, 2004.

[GBBZ03] S. Greiner, B. Boskovic, J. Brest, V. Zumer. Security issues in information systems based on open source technologies. *EUROCON*, 2003.

[HS02] 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.

[KENU07] 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.

[Lau09] LaunchPad Home page. https://launchpad.net/, 2009. https://launchpad.net/

[MFH99] 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.

[MFH02] 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.

[MHP07] 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.

[Mic05] M. Michlmayr. Software Process Maturity and the Success of Free Software Projects. *Software Engineering: Evolution and Emerging Technologies* 130:3–14, 2005.

[Mic07] M. Michlmayr. *Quality Improvement in Volunteer Free and Open Source Software Projects – Exploring the Impact of Release Managemen*. PhD thesis, University of Cambridge, UK, 2007.

[OMK08] 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.

[RFL05] 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.

[RM02] 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.

[Rob02] 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.

[SC09] 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.

[Sca03] 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.

[Sca07] W. Scacchi. Free/Open Source Software Development : Recent Research Results and Emerging Oppurtunities. *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.

[SFF+06] 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.

[Sou09] SourceForge Home page. http://sourceforge.net/, 2009. http://sourceforge.net/

[Sta02] J. Stark. Peer reviews as a quality management technique in open-source software development projects. *European Conference on Software Quality*, pp. 340–350, 2002.

[YK] 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.

[ZE00] L. Zhao, S. Elbaum. A survey on quality related activities in open source. *ACM SIGSOFT Software Engineering Notes*, pp. 53–57, 2000.

[ZE03] L. Zhao, S. Elbaum. Quality assurance under the open source development model. *The Journal of Systems and Software* 66:65–75, 2003.