Overview on Trust in Large FLOSS Communities

Etiel Petrinja, Alberto Sillitti, and Giancarlo Succi
CASE – Center for Applied Software Engineering
Free University of Bolzano/Bozen
Piazza Domenicani, 3
I-39100 Bolzano-Bozen,
{Etiel.Petrinja|Alberto.Sillitti|Giancarlo.Succi}@unibz.it
WWW home page: http://www.unibz.it

Abstract. The paper presents a survey of mature Free/Libre Open Source Software communities. The main focus of the survey is the collection of data related to the practices of these communities related to trust elements in their products. The survey is carried out using a structured questionnaire about thoughts and practices followed by Free/Libre Open Source Software communities. The survey focuses on the analysis of the development processes adopted by such communities. The results of the survey confirms basic ideas related to Free/Libre Open Source Software and explains in more detail specific issues related to trust and trustworthiness of the Free/Libre Open Source Software development process.

Keywords: FLOSS Communities, Software Quality, FLOSS Development Process

1 Introduction

The survey presented in this paper analyses a set of Free/Libre Open Source Software (FLOSS) projects that are used by large communities. The survey focuses on issues related to the trust that users have in adopting FLOSS products. We are interested mainly in processes, tools, methods, and approaches adopted to develop FLOSS products. The survey is part of a four year EU funded project named QualiPSo (Quality Platform for Open Source Software) aiming at studying methodologies, technologies, and policies to leverage the Open Source Software development to sound, well recognised, and established industrial operations [8].

In the last decade, FLOSS products are increasingly being used. This increase is supported by several factors including the absence of direct license costs and the availability of the source code that allows users to adjust FLOSS products to their specific needs. An important drawback of FLOSS is the lack of quality assurance metrics that prove its validity. In commercial environments, the problem is addressed

---

Please use the following format when citing this chapter:

Petrinja, E., Sillitti, A. and Succi, G., 2008, in IFIP International Federation for Information Processing, Volume 275; *Open Source Development, Communities and Quality*; Barbara Russo, Ernesto Damiani, Scott Hissam, Björn Lundell, Giancarlo Succi; (Boston: Springer), pp. 47–56.

in a different way. The concept of quality is often related to the certification of the production process of the producer (e.g., CMMI). This approach can be extended also to the FLOSS production (in particular, for FLOSS developed supported by companies). The survey presented in this paper is trying to find out which elements contribute to the trustworthiness that people have in FLOSS products. Trust is linked to both characteristics of the final product and the overall development process followed. In particular, the process is an element that may vary considerably among different FLOSS projects. The survey collects information about different approaches and synthesises them to identify benefits and avoid pitfalls of FLOSS development.

2 Background

In the last decade, intensive research has been carried out on FLOSS. Benchmark results have provided a list of characteristics that are common in the FLOSS development process. The most important are the following [1]:

- it is parallel, rather than linear,
- it involves large communities of globally distributed developers,
- it utilizes truly independent peer review,
- it provides prompt feedback to user and developer contributions,
- it includes the participation of highly talented, highly motivated developers,
- it includes increased levels of user involvement, and
- it makes use of extremely rapid release schedules.

Moreover, the whole FLOSS life cycle differs from the classical software development processes. Mockus *at al.* [7] and Jorgensen [4] proposed similar models identifying phases that are present in FLOSS development projects. The Jorgensen's model [4] includes the following list of phases: code, review, pre-commit test, development release, parallel debugging, and production release.

FLOSS development is strongly influenced by software development tools used by both developers and contributors. In contrast to the diffusion of Computer Aided Software Development (CASE) tools used by traditional software development, FLOSS process adopted software tools as issue/bug trackers, mailing lists, forums, collaboration environments, etc. An important collaborative development environment – SourceForge [3] – provides an integrated environment where more than 100.000 FLOSS projects are being stored and developed (even if only a small portion of them are active projects involving several developers). FLOSS uses also traditional software engineering tools, however not all tools are available as FLOSS products. Therefore, in the near future these tools will eventually appear as FLOSS and will further improve the FLOSS process.

Many researchers have studied the quality and trustworthiness of FLOSS products and their development processes. Since quality is a fundamental ingredient of software and a relevant criterion for adopting FLOSS products, the research was oriented toward the evaluation of the current quality of FLOSS products and how to improve it. Lipner [5] explored the benefits and possible pitfalls of FLOSS development. McGraw [6] stated that "openish" products will not improve security of the software. Schneider [9] stated that source code inspections are just one of possible approaches to improve software's quality by discovering bugs. Witten [11] explored economics of FLOSS products, metrics used to assess it, and models available.

3 Research Design

The survey includes two parts: one related to FLOSS products and the other related to FLOSS processes. Results presented in this paper deal only with the second part of the questionnaire.

4 Scope

The survey includes seven well-known FLOSS projects: Apache HTTP Server, Eclipse, Emacs, Linux Kernel, Mozilla project, GNOME, and Debian.

5 Methodology

The survey uses the approach proposed by Silverman [10]. It requires the design of a structured and formal research involving two basic and partially correlated concepts:

- The methodology, that is, the specific technique for gathering data (survey, interview, questionnaire, case study, etc.).
- The method, that is, whether performing a quantitative or a qualitative investigation.

Such decisions are based on an evaluation of the goals of the research and the kind of information required.

Our study focuses on gathering opinions about the FLOSS process and products adopted in the surveyed communities. Therefore, we have to conduct a qualitative investigation that involves the analysis of data such as words and sentences instead of numbers [10]. Our research methodology is based on a semi-structured questionnaire filled in mainly by analysing projects' web sources such as web pages, CVS repositories, mailing lists, and forums. Moreover, some data comes from face-to-face or telephone interviews.

5.1.1 GQM

The overall structure of the research is based on the GQM approach [2], as follows.

- Goal: Evaluate the actual adoption of FLOSS in the software industry.
- Question: The questionnaire is composed of 53 questions with additional sub-questions.
- Metrics: Metrics about the level of adoption and the trust in FLOSS products.

5.1.2 Questionnaire

The questionnaire is organized in 16 sections dealing with different topics related to the FLOSS development processes. The 16 topics are the following:

1. Personal information
2. Company information
3. Role of the organization with respect to FLOSS
4. Issues that can be taken into account when deciding whether to adopt FLOSS
5. Trust
6. Quality assurance
7. General questions
8. Roles and responsibilities
9. Architecture definition
10. Development techniques and practices
11. Tools used
12. Features to implement
13. Documentation and bug management
14. Version control and people management
15. Business model
16. Workflows of the processes identified

The answers gathered through face-to-face interviews are collected following the following steps:

- The respondents are contacted to determine their general interest in the study.
- The questionnaire is sent to the respondents to verify the actual availability.
- Data is collected by personal or telephone interviews.
- The results of the interviews are sent to the interviewees for a final check.

Only upon a positive feedback from the interviewee, the questionnaire is considered accepted and the data is processed. Participants are guaranteed anonymity and the information reported is reviewed so that no individual person or company can be identified. The final questionnaire was designed iteratively; during the survey we added just a small number of new questions and in a few cases we have changed the order of questions to improve the focus on specific topics.

6 Results

In the following subsections we present four topics of the questionnaire: quality related issues, stakeholders related issues, technology related issues and business related issues.

6.1 Quality issues

6.1.1 Trust

Important elements to people surveyed are: openness of the whole development process, openness of the planning process, testing and integration builds, presence of intermediate milestones and visibility of the planed and the actual development process. Moreover, communities trust more common elements such as: the quality of the source code, the correct behaviour of the product, its performance, and often the security of the developed product. We have found out that communities try to fulfil classical trust-related criteria first and additionally they attempt to satisfy important FLOSS-related criteria as well. The same is true when they are testing external FLOSS products to be adopted by the community.

6.1.2 Quality assurance

Quality elements considered important by FLOSS communities vary considerably among different communities. The most important elements listed by communities are the following: the planning process used, the development process followed, the compatibility of the licenses used by specific subprojects, the bug/issue reporting and solving procedure, the availability of appropriate documentation, the simplicity of installation, and a proper integration of different subparts of the final product.

Answers related to testing processes of FLOSS products, either developed by the community or adopted by it, is quite homogeneous among the considered communities. The majority bases the testing process on users and developers that produce FLOSS products. Almost half of the communities have specific test teams that provide a defined quality level of the developed products. On the contrary, one quarter of the communities said explicitly that they do not use any specialized test team. Part of the communities employs beta testers selected from the group of their regular users that provide useful bug/issue reports before the product is largely distributed to the public (Fig. 1).

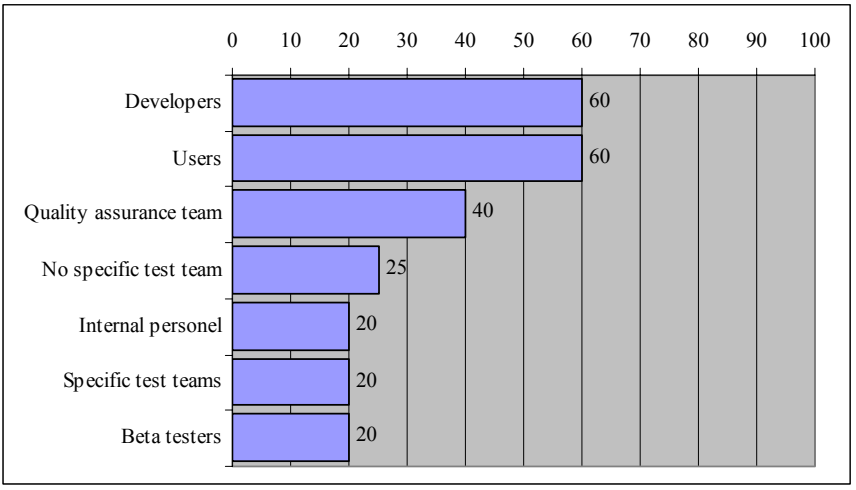

Fig. 1. Percentage of different stakeholders testing FLOSS products.

6.2 Stakeholders related issues

6.2.1 Roles and responsibilities

The number of developers involved in each surveyed FLOSS project varies from 25 core developers to more than 1000 developers. The roles represented in these communities are: simple users, developers, committers, and Project Management Committee (PMC) members. Communities have often an additional management body as can be a technical or a non-technical board that reviews the work done by the community and the progress of the project. Half of the communities surveyed are also supported by a foundation that manages the project and overviews the alignment of the project evolution with basic directives that the project has to satisfy. There is not a common hierarchy structure in different communities. Usually, FLOSS communities do not have many hierarchy levels and they tend to implement a democratic system where everybody has the possibility to express his opinion.

Roles are not always strictly fixed and users can obtain more privileges by providing good quality contributions, stay aligned with local policies and written and not-written rules. In some communities, users can become developers if they provide good quality code and can become committers if their contributions are significant. The role of a user depends on the definition of specific roles in different communities. Often the PMC is responsible for granting new roles and privileges.

6.2.2 Features to implement

New features to be implemented in FLOSS projects are usually proposed in mailing lists or in bug/issue management systems. Features are usually not ranked in the surveyed communities. Exceptions are some bug fixes and features related to impellent architecture modifications. In some communities, new features can be

implemented immediately by developers and contributors, in others, more structured communities, these suggestions can be included in the future implementation plans and roadmaps. The majority of surveyed communities have a time plan for new features to be implemented in the following few months. Usually, plans are available also on web pages to allow everybody to see which features will be added in the near future. In this way, users can participate to the implementation process with source code or documentation contributions.

When changes are proposed, the PMC, the supporting foundation or the responsible person for a specific module decides which features will be implemented. Usually, in the more hierarchically structured communities surveyed there is a specialized development team that implements new features. This team is composed by developers and committers that are responsible to the PMC or to the owners of specific modules. In less hierarchical communities, developers and committers may decide which new features they would like to implement. The possibility to choose which features a developer will implement can be an important productivity advantage that FLOSS processes have in comparison to the proprietary ones.

6.3 Technology issues

6.3.1 The overall architecture

The architecture of the system is defined incrementally or, in few cases, it is planed from the beginning. This depends on the nature and on the size of the project. If projects are strongly centralized they usually have a well planed architecture; on contrary, if projects are a collection of smaller modules, the architecture of the system is defined by just few leading rules. In the majority of the surveyed communities, the architecture planning is often the combination of both approaches.

FLOSS projects are often based on very specific standards. It is the case of the Apache HTTP web server that implements the HTTP open standard. Standards implemented are in the majority of the cases open and sometimes also supported and proposed by FLOSS communities. Another important element present in the surveyed FLOSS communities is the interoperability of the software developed with other (FLOSS and commercial) products. Open standards and interoperability issues are usually interconnected.

6.3.2 Tools used

The operating system used to develop FLOSS projects by all the surveyed communities is either Linux or a Unix-like operating systems. Since the projects surveyed have started many years ago, the products have been adapted to several popular operating systems. More than half of them can run on Microsoft operating systems and many on MacOS.

The most frequently used programming language is not Java as we expected from the current mainstream FLOSS development, but C/C++. The main reason for this is the longevity of the projects surveyed that started in the ‘90s when Java was

not yet so popular. However, the surveyed communities often use also other languages such as C#, Perl, Python, Lisp, and Java.

The number of different programming tools used by the surveyed communities is very large. One FLOSS community reported that they use 820 different tools and libraries for their development. However, the tools used by the majority of the communities are source code management tool such as CVS (but also GIT, Bit Keeper, LXR, etc.), bug/issue tracking tools such as Bugzilla, and mailing lists .

6.3.3 Development techniques and methodologies

Usually, the adopted development techniques are not well described by the surveyed communities. However, they often explicitly write guiding principles that are used inside the community. Such principles are for instance Quality Culture, Collective Reputation, Freedom, Autonomy, and Evolution. Another element present in the majority of communities is the development process divided into distinct phases that are clearly defined by the community. Transitions from one phase to another are open and there are public reviews. Common phases are: Pre-proposal, Proposal, Incubation, Mature, Top-Level, and Archived.

6.3.4 Documentation

A detailed documentation is a very important part of all the projects developed by the surveyed FLOSS communities. Usually, they start a subproject that is responsible of documentation. It can be produced in different forms, most often as user manuals (separate documents), documentation inserted inside the code (JavaDoc or similar), developer's and maintainer's manuals, and web pages. The creation and the maintenance of the documentation are open to everybody willing to contribute with some effort. Contributors can come also from persons that are not programmers. Documentation is usually protected by a FLOSS consistent license such as Creative Commons.

6.3.5 Bug/Issue management

All the communities surveyed have mailing lists focused on bugs. The majority of communities have also an automatic bug/issue tracking system that helps users and developers to report and manage bugs. More than half of the communities use Bugzilla as the FLOSS bug/issue-tracking solution.

The bug solving procedure depends on the severity of the bug reported, the size of the project, the specific community, and some other issues related to specific bug. The time needed to solve a bug inside communities varies considerably: from one day for 75% of the bugs, up to 140 days for 90% of the bugs in the Apache Server community. Other communities report longer response times for not critical bugs. However, the majority of communities try to provide some answer to critical bugs in one to three days from the notification.

6.4 Business related issues

The business model driving the FLOSS movement does not include revenues obtained from selling licenses for the products. However, there are many additional services, courses, publications, and adaptations that are offered to the market by companies that in many cases collaborate with FLOSS communities. Moreover, there are many indirect benefits that are offered to contributors. Some of them are: the improvement of the reputation of developers, the possibility to get better job positions, working with people that share the same ideas, advancing of FLOSS software in comparison with proprietary software, etc. Some FLOSS communities distribute also grants and prizes that are offered by supporting companies.

A very important aspect that has emerged from various researches done in the last decade on the FLOSS movement, and also from our survey, is the support offered by large software companies to the FLOSS movement. Key developers and leaders in FLOSS communities are often employees of world leading IT companies such as IBM, Hewlett-Packard, Intel, Red Hat, Sun Microsystems, etc. Developers are paid to work on the development of FLOSS products. Companies supporting FLOSS development receives benefits from other sources such as: publicity obtained by contributing to the community, a deep knowledge of FLOSS that can be sold along with proprietary products, attracting good young developers, etc.

7 Conclusions

The results of the survey confirm our expectations on the most important trustworthy elements perceived by FLOSS products developers and users as: the number of downloads, the longevity and the level of activity in the community. The survey revealed additionally more in details which characteristics are essential for FLOSS communities to trust external FLOSS products. The most important aspects of FLOSS development are, as expected, the openness of the whole development process and continuous testing of the product. FLOSS communities try to fulfil first important generic requirements that are guiding also proprietary software development, and additionally they try to accomplish also specific FLOSS related requirements.

Important elements that proof the quality of FLOSS products are: the license used, the quality and completeness of the documentation, and a thorough testing. FLOSS communities often base their testing on specific groups of developers but they rely especially on the large community of users that is an essential part of each FLOSS project. The survey has confirmed also a growing importance that have traditional world leading software companies in further growing of the FLOSS movement. Companies contribute intensively to FLOSS communities usually by paying developers that work for the community. Therefore, many trust related approaches and procedures are often migrated from companies to FLOSS communities.

Standard software development techniques combined with essential FLOSS principles form a higher quality and trustworthy hybrid software development approach. These changes will eventually improve the credibility of FLOSS products and increase their use in companies and public administrations.

8 Acknowledgment

The authors would like to thank the partners of the QualiPSo project that have been involved in the survey. In particular, Prof. Sandro Morasca of the Università dell'Insubria (Italy) who participated to the design of the questionnaire, Prof. Jesus M. Gonzalez-Barahona and Prof. Gregorio Robles of the University Rey Juan Carlos (Spain) who collected the data about the GNOME and Debian communities.

9 References

[1] The Leading Linux Resource in the World! http://linux.sys-con.com/, Last visit December 2007.
[2] V. R. Basilli: Software modeling and measurement: The Goal/Question/Metric paradigm. Department of Computer Science, University of Maryland, 1992.
[3] J. Howinson and K. Crowston: The Perils and Pitfalls of Mining SourceForge. University Academic Department.
[4] N. Jorgensen: Putting it All in the Trunk, Incremental Software Development in the FreeBSD Open Source Project. Information Systems Journal, (2001), 11, 321-336.
[5] S. B. Lipner: Security and source code access: Issues and realities. University Academic Department, 124-125.
[6] G. McGraw: Will openish source really improve security? University Academic Department, 128-129.
[7] A. Mockus, R. Fielding and J. Herbsleb: A Case Study of Open Source Software Development: The Apache Server. University Academic Department, 263-272.
[8] QualiPSo - Quality Platform for Open Source Software. http://www.qualipso.org/index.php, Last visit December 2007.
[9] F. B. Schneider: Open source in security: Visiting the bizarre. University Academic Department, 126-127.
[10] D. Silverman: Doing qualitative research. Sage Publications, 2000.
[11] B. Witten, C. Landwehr and M. Caloyannides: Will open source really improve security? University Academic Department.