Towards The Evaluation of OSS Trustworthiness: Lessons Learned From The Observation of Relevant OSS Projects

Davide Taibi[1], Vieri del Bianco[1], Davide Dalle Carbonare[2], Luigi Lavazza[1], and Sandro Morasca[1]
[1] University of Insubria
davide.taibi | vieri.delbianco | luigi.lavazza | sandro.morasca@uninsubria.it
WWW home page: http://www.uninsubria.it
[2] Engineering Ingegneria Informatica S.p.A.
davide.dallecarbonare@eng.it
WWW home page: http://www.eng.it

Abstract. To facilitate the adoption of open-source software (OSS) in industry, it is important to provide potential users (i.e., those who could decide to adopt OSS) with the means for evaluating the trustworthiness of OS products. This paper presents part of the work done in the QualiPSo project for this purpose. A set of factors that are believed to affect the perception of trustworthiness are introduced. In order to test the feasibility of deriving a correct, complete and reliable evaluation of trustworthiness on the basis of these factors, a set of well-known OSS projects have been chosen. Then, the possibility to assess the proposed factors on each project was verified: not all the factors appear to be observable or measurable. The paper reports what information is available to support the evaluation and what is not. This knowledge is considered to be useful to users, who are warned that there are still dark areas in the characterization of OSS products, and to developers, who should provide more data and characteristics on their products in order to support their adoption.

Keywords: OSS trustworthiness, OSS quality, OSS adoption

1 Introduction

The success of OSS is due to multiple reasons, ranging from technical qualities to financial, ethical and political motivations. Nonetheless, the adoption of OSS is still limited. The reason is that, in several cases, OSS fails to convince potential users that its adoption is safe and poses no more risks than purchasing commercial software. In this paper, we report on the initial work, carried out in the QualiPSo project, that focuses specifically on the characteristics of OSS products and artefacts, in order to identify the ones most closely related to trustworthiness. The QualiPSo (Quality Platform for Open Source Software) project 0 is an ongoing initiative that proposes a

Please use the following format when citing this chapter:

Taibi, D., del Bianco, V., Carbonare, D.D., Lavazza, L. and Morasca, S., 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. 389–395.

coherent and systematic evaluation of the trustworthiness of OSS projects, and aims at promoting the diffusion of OSS by focusing on OSS trustworthiness.

We name "trustworthiness" the set of qualities that are of interest for the users, especially in the process of deciding whether a given OS program (or library, or other piece of software) is "good enough" to be used in an industrial or professional context.

Firstly, we defined the set of factors that were believed to be the most closely related to the perceived trustworthiness 0; then we identified a set of OSS projects, widely adopted and generally considered trustable, to be used as references. Afterwards, a first quick analysis was carried out, checking which factors were readily available on each project's web site. The idea was to emulate the search for information carried out by a potential user, who browses the project's web sites, but is not willing to spend too much effort and time in carrying out a complete analysis. Since the view of trustworthiness factors emerging from the analysis seemed too subjective, it was decided to precisely define measures specifying how to evaluate the OSS characteristics, and how to collect data that could be effectively used in the analysis phase, to be performed according to some statistical methods.

2 Project Selection and Analysis

The selection of projects addressed different types of software applications, generally considered stable and mature. The complete set of projects comprises 32 products, different with respect to age, implementation language, size of developers and users communities, etc.

Here the criteria used to select a representative set of OSS projects are reported. Projects have a set of characterising attributes. The selection criteria aimed at:

- Including a reasonably small set of projects.
- Including at least a couple of projects for every possible value of any attribute.

For instance, an attribute is the size of the development team. Four possible values were defined: 0 (inactive project), no more than ten people, up to 50 people, more than 50 people. Therefore, we took care to include at least two projects for each of the four mentioned classes. The complete set of attributes is reported in Table 1.

Table 1. The projects' attributes.

Attribute	Possible values
Repository	SourceForge, Apache, Java.net, FreshMeat, Rubyforge, ObjectWeb, Free Software Foundation, SourceKibitzer, other
Standalone	Yes/No (Part of a Project family)
Type	Web Server, Operating System, ERP, CSM, …
Developer organization type	Sponsored/foundation, spontaneous, other
Cost	Free; pay for services and features; pay for everything
Size of the development team	0 (abandoned/closed project), 1–10, 11–50, >50

User community size	Small (<51), Medium (51–250), Large (>250)
Programming language	Java, C#, C/C++, scripting languages, Visual Basic, other
Tool support(*)	little use of tools (0-4 tools used); extensive use of tools (5-7)
Innovation	Traditional application (existing before 2003); Emerging application (only proprietary solutions before 2003)
Age	Project started before 1998; between 1998 and 2003; after 2003

(*) the potentially supported activities are: continuous integration (supported by Cruise Control, Damage Control, Continuum, …), building (Ant, make, …), code documentation (Doxygen, Javadoc, …), testing (JUnit/NUnit/PhpUnit, …), version control (CVS, Subversion, …), bug tracking (Jira, Bugzilla, …), static code analysis (Spoon, Checkstyle,…).

We analyzed the 32 selected projects, considering the information concerning the trustworthiness factors. The analysis was carried out by looking for information that was readily available in the project web sites.

Table 2 lists the factors that are believed to determine trustworthiness, and reports, for each factor, how many projects provided (in the official web site) enough information to evaluate the factor.

Table 2. Number of projects that provide data about the considered factors

Factor	N° of projects supporting the evaluation of the factor
Type of licenses used	32
Facilities for developing modifying...	10
Technical manual	2
Mid/long term existence of a user community	Not Found
Existence of a sufficiently large community	Not Found
Short term support	Not Found
Mid/long term existence of a maintainer organization	Not Found
Use of standard architecture	11
Usage of patterns	6
Programming language uniformity	25*
Complexity	Not Found
Size	Not Found
Reliability	8
Maintainability	7
Modularity	18
Usability	9
Portability	15
Performances	8
Functional requirements satisfaction	11
Customer satisfaction	Not Found
Interoperability with external systems/integration ease	10

Availability of user manual	30
Standard compliance	10
Availability of best practices	17
Human interface language/localization	15
Self containedness	15
Existence of benchmarks or test suites that witness quality	5
Availability of training, guidelines, use cases, tutorial	20
Distribution channel	31
Number of downloads	32

* Only a few of these projects explicitly declare to use one language for the whole project.

3 Factor Refinement

The experience of the quick project analysis showed that for several factors it was necessary to define more precise and specific measures. The need to base evaluations on more objective data also emerged. Accordingly, whenever a factor proved not to be directly measurable, a set of 'proxies' was defined. Some proxies can be assessed in a simple and direct manner, while others need specific tools. Table 3 reports both the new measures and the unchanged ones defined for OSS product trustworthiness. The idea is that for each OSS project the factors are evaluated according to these measurement definitions.

Table 3. New criteria for the evaluation of trustworthiness factors

Factor	Basic indicators
Functional requirements satisfaction degree	Availability of: feature list, free text description, release notes, product example/demo
Customer Satisfaction	List of organizations, testimonials and other projects using this software, case studies, usage histories User community satisfaction (according to forums, blogs, mailing lists, newsgroups, magazine/scientific articles)
Interoperability	Communication with other systems supported by suitable mechanisms (SOAP, Web services, protocols, public interfaces, …); Ease of integration with other products and possibility to migrate to other product with little effort
Reliability	Development status, frequency of patches, average bug time solving
Maintainability	Existence of a guide to extend/adapt the OSS product, maintenance releases and architectural documentation. Coherent usage of coding guidelines/standard, source code quality and programming language uniformity
Modularity	The product provides plug-in interface(s)
Standard Architecture	Availability of architectural documentation and usage of architectural standard/pattern

Mid Long Term Existence of a User Community	Project Age; Trend of the number of users; Number of patches/releases in the last 6 month; Number of developers involved; Average bug solving time
Availability of technical and user documentation	Availability of: up to date technical/user manual, getting started guide, installation guide, Technical/User related F.A.Q., Technical/user forum and mailing list
Standard Compliance	Any information about standard implemented (like HTTP 1.0, SQL 97...) and coding standards
Existence of a sufficiently large community of users	Number of posts available on forums/blogs/newsgroups and related activity
Performance	Existence of performance tests and/or scenarios, specific performance-related documentation Implementation -Any best practices, concerning design and product construction, aimed at boosting performance.
Type of License	Main and sub license used
Short Term Support	Bug number, bug removal rate, availability of professional services
Availability of facilities for developing, modifying, and customizing the product	General purpose build tools applicable to the product, build script, built-in customization facility (configuration API, …)
Usability	Detailed feature description and user manual Ease of installation/configuration, ease of use.
Portability	Supported environments, usage of a portable language (like Java), environment-dependent implementation (e.g., usage of hardware/software dependent libraries)
OSS Provider Reputation	Opinion and feedback from other users
Best Practices	Availability of best practices, code examples/tutorials
Programming language uniformity	Number of languages used in the project
Complexity	McCabe complexity number or any related information available on the web site
Human Interface Language Localization	Localization support availability (e.g., are language files provided?)
Self Containedness	Can the product be installed and executed “out of the box” or does it require other software? Are dependencies documented?
Existence of benchmarks/test suites that witness for the quality	Availability of test suites/benchmarks, Usage of a test framework (JUnit, DejaGNU,…), results of tests published (on the project site), existence of initiatives to encourage the community to contribute to quality efforts
Mid/long term existence of a maintainer / sponsor	Active maintainer organization / sponsor
Availability of training, guidelines, use cases, tutorial etc.	Up to date training materials, manuals and guidelines available free of charge Availability of official training courses
The distribution media	Source code download; Binaries download Access to the project repository; CD/DVD distribution
Size	Number of Lines of code, source files and functions (or classes

	and methods, for object oriented code)
Popularity of the product	Number of downloads

The main areas that could not be covered in the previous analysis were those related to the quality of the product and the user community. In no project website we found any indication about the user community size, the internal software quality and complexity, or the vitality of the project.

Proxy definitions were conceived to make the assessment as easy and objective as possible. The possibility of employing tools –possibly specifically developed for this purpose– was also taken into account. Considering for instance the evaluation of *the mid/long term existence of a user community*, we suggest checking the following indicators: the growth rate of the community, the number of patches/releases during the last 6 months, the number of developers involved, the average bug solving time and the project age. In order to assess the growth trend of the user community we shall develop a specific tool, which –by digging into the web portals, official forums and blogs– computes the number of unique users in the last months.

Unfortunately, some information considered important is never exposed on the project websites; therefore we wish to throw a suggestion to the Open Source community, recommending the leaders of OSS projects who would like to publicize the trustworthiness of their products to publish all the useful data. In any case, there are some factors that are inherently difficult to evaluate: for instance, it is quite hard to evaluate the quality of the supplied user manual. This task could be made much easier if it were possible to collect feedback from users: it is thus important to make the users aware that the usage of some feedback collector would be beneficial to the whole user community.

4 Conclusions

In order to favour the adoption of OSS, it is necessary to assure the potential users that the OSS products do meet their expectations under several respects. In the QualiPSo project, the notion of "trustworthiness" is meant to include several qualities of the OSS, ranging from training support to the possibility of modifying/adapting the programs, to the availability of support from the producer, etc. Since the notion of trustworthiness is quite broad, it is necessary to fully understand what factors contribute to making a product trustworthy. For this purpose, in the QualiPSo project several OSS products have been examined: a set of factors that are believed to affect the perceived trustworthiness were identified, a set of outstanding OSS project were selected and are being analysed with respect to the mentioned factors.

Here we reported the preliminary results of the analysis of OSS products and artifacts. Next steps will include a second analysis round, based on the usage of the newly defined measures, possibly along with a campaign, addressed to OSS

developers, to provide more information on the characteristics that affect the trustworthiness of their products. A number of OSS code repositories will also be analyzed, with the aid of automated tools.

The collected data will be analyzed, with the objective of identifying commonalities and differences in the characteristics and usage of OSS products, to prepare the ground for the creation of a trustworthiness model encompassing the characteristics and factors that have been observed to actually affect the perception of trustworthiness in OSS products and artifacts.

Although the analysis is not yet complete, we were able to make some preliminary observations. A first result is that by browsing the information provided by the projects' web sites, only a relatively small set of the interesting factors could be evaluated: several factors appear not to be observable or measurable. The paper reports what information is not available: developers should provide the missing information in order to support the adoption of their products. Since some of the trustworthiness factors could not be evaluated because of their subjectivity, we began a more precise definition of the measures – illustrated in Table 3 – that should be used to capture these factors.

References

1. V. del Bianco, M. Chinosi, L. Lavazza, S. Morasca, D. Taibi, "How European software industry perceives OSS trustworthiness and what are the specific criteria to establish trust in OSS", October 2007, available on-line at http://qualipso.semanticdesktop.org/xwiki/bin/view/Wiki/WP51.
2. QualiPSo project web site: http://www.qualipso.eu