QualiPSo

Quality Platform for Open Source Software

IST- FP6-IP-034763

Deliverable D5.1.1

How European software industry perceives OSS trustworthiness and what are the specific criteria to establish trust in OSS

Vieri del Bianco
Michele Chinosi
Luigi Lavazza
Sandro Morasca
Davide Taibi

Due date of deliverable: 31/10/2008

Actual submission date: 06/11/2008

This work is partially funded by EU under the grant of IST-FP6-034763.

Change History

Version	Date	Status	Author (Partner)	Description
02.01	20/10/2008	draft	INS	
02.02	28/10/2008	release candidate	INS	

EXECUTIVE SUMMARY

The goal of this document is to gather and summarize information about the trustworthiness goals of organizations (Private, Public Administrations, No Profit, Education Institutions, etc.) when they deal with (develop, use, integrate, customize, etc.) Source Software (OSS), and to summarize the factors that influence their decisions.

The information provided by the various OSS players is evaluated and compared in order to assess commonalities and differences between their viewpoints. The results of the analysis presented in this document will influence all subsequent work in QualiPSo's Activity A5. Specifically, the results reported here will be used as a basis for defining a set of measures to capture the trustworthiness of OSS products, a set of measures to capture the factors that may influence trustworhiness, and a set of models that link these influencing factors to trustworthiness in such a way that they are relevant and useful to the European software industry.

Information gathering has been carried out through questionnaires. The answers have been analyzed to extract the most significant trustworthiness factors.

Some of the results found were somewhat expected, and are aligned to literature findings, like the high importance given to user requirements, interoperability, standard compliance, and to the community in general. External software qualities and documentation are considered very important when choosing a OSS project.

However, other results contradicted our expectations as well as the literature, and will need further analysis: for instance, product size and complexity turned out to be considered of low importance in our results, while these attributes are generally accepted as relevant and widely used in the characterization of software products. Also, economic factors and licenses were considered of a somewhat lower importance than expected.

After carrying out an initial set of 103 interviews, data collection is ongoing and further results have been obtained. However, the vast majority of the new results seems to confirm the initial results.

Document Information

IST Project Number	FP6 - 034763	**Acronym**	QualiPSo
Full title	Quality Platform for Open Source Software		
Project URL	http://www.qualipso.org		
Document URL			
EU Project officer	Charles MacMillan		

Deliverable	**Number**	D5.1.1	**Title**	How European software industry perceives OSS trustworthiness and what are the specific criteria to establish trust in OSS
Work package 5.1	**Number**	5.1	**Title**	Goal Information Gathering
Activity A5	**Number**	5	**Title**	Trustworthy Results

Date of delivery	**Contractual**	31/10/2008	**Actual**	06/11/2008
Status	Version 02.02, dated 28/10/2008		final □	
Nature	Report ☑ Demonstrator □ Other □			
Dissemination Level	Public □ Consortium ☑			
Abstract (for dissemination)	In the last few years, Open Source Software (OSS) has been widely accepted and adopted, but there still are several concerns on how much trustworthy OSS actually is, and how its trustworthiness can be evaluated. These doubts led to a much lower adoption rate than expected. We are interested in what OSS users think about OSS trustworthiness, so we can then devise evaluation methods for OSS that are useful to users in general and the European software industry in particular. The objective of the investigation documented in this report is twofold: 1) to understand the reasons and motivations that lead software companies to adopt or reject OSS and software developers to develop OSS, and 2) to understand which specific trust factors are taken into account when selecting an OSS product. We interviewed quite a few representative industrial organizations and several people with various professional roles. Here, we report the insights in OSS trustworthiness deriving from the statistical analysis of the results of the interviews, as well as on the review of the relevant literature.			
Keywords	Trustworthiness, OSS, Empirical Software Engineering, Statistical analysis			

Authors (Partner)	Vieri del Bianco (INS), Sandro Morasca (INS)			
Responsible Author	Vieri del Bianco		**Email**	vieri.delbianco@uninsubria.it
	Partner	INS	**Phone**	+39 0332218938

TABLE OF CONTENTS

1 INTRODUCTION

In the last few years, Open Source Software (OSS) has been widely accepted and adopted. Some well known success stories should dispel doubts on the potential of OSS when compared to non-free or closed software[1]. As a matter of fact, OSS usage is constantly growing in every ICT sector, but there still are lots of concerns on how trustworthy OSS actually is and the trustworthiness of OSS can be evaluated. As a consequence, this has led to a much lower adoption rate than could be expected.

Hence, we are interested in what OSS users think about OSS trustworthiness, so we can later devise evaluation methods ways for it that are useful to the European software industry. By the generic term "OSS users," we mean all the types of users of an OSS product, including developers, integrators, system administrators, product managers, clearing house members, end users, etc.

The objective of the investigation documented in this report is twofold:

- To understand the reasons and motivations that lead software companies to adopt or reject OSS, and, symmetrically, software developers to develop OSS. In other words, we want to investigate the reasons behind the demand and the supply of OSS. Specifically, we focus on the trustworthiness of OSS, since OSS users will not adopt a specific OSS product unless they can trust it. On the other hand, OSS developers need to promote the trustworthiness of their products, so that they may be more appealing to end users.
- To understand which specific trust factors are taken into account when selecting an OSS product. It is expected that a long list of factors may be identified when taking into account many viewpoints, given the inherent complexity of the problem at hand and the large amount of OSS communities and products, each of which can target specific application domains and have different organizational structures and sizes. Thus, we are interested in prioritizing such factors, to find if there are any common trends that may be identified across different users and different domains. As a consequence, we are also interested in investigating whether some indicators may be better suited for specific application domains or for specific types of software companies.

In QualiPSo's activity A5, we have adopted a goal-oriented approach, in which we first target the software industry's needs and beliefs. The knowledge of these needs and beliefs will be used in the subsequent work in activity A5 to derive a set of indicators that can be used to assess OSS trustworthiness. A number of indicators and OSS evaluation models are currently available, but it is unclear whether they really respond to industrial needs and whether they are based on industrial beliefs. Some of these indicators are based on "traditional"

[T] To mention only a few representative projects: Linux kernel (http://www.kernel.org), MySQL (http://www.mysql.com), Apache forge (http://www.apache.org), Eclipse (http://www.eclipse.org), Mozilla (Firefox and Thunderbird) (http://www.mozilla.org), PHP (http://www.php.net),Python (http://www.python.org),Ruby and Ruby on Rails (http://www.ruby-lang.org, http://www.rubyonrails.org),Jboss (http://www.jboss.com), etc.

software measurement, while others have been (at least initially) defined for specific needs: for example, we have metrics for Object Oriented systems, quality factors for web applications, reliability factors for component based architectures, security factors for applications that manage sensible data, process quality factors to assess development processes, etc.

To this end, we have carried out a number of interviews to elicit these goals and factors directly from industrial players, as we focus on industrial contexts. This report documents and summarizes the information we have gathered from the set of interviews we have carried out.

We designed a questionnaire and interviewed quite a few representative industrial organizations and, within them, several people with various professional roles. Our belief is that it is much better to ground the derivation of indicators for assessing the trustworthiness of OSS products on real industrial needs, than to derive them abstractly from our own personal beliefs and/or by reading the available literature. We are certainly aware that it would be impossible to capture the goals and factors of interest of every OSS user. In addition, even within the same software organization, different goals and factors co-exist, due to the different roles that people have in their organizations.

After an initial set of 103 interviews, we continue to interview OSS stakeholders, by 1) in person interviews with a refined questionnaire, in which most of the open questions were closed, and 2) via an online version of the questionnaire. Open questions could be closed becaus the initial set of interviews has allowed us to identify the factors that were deemed relevant by OSS stakeholders. This allowed us to somewhat reduce the time needed for the interviews and build the online questionnaire. We have collected 48 additional questionnaires, whose results mostly confirm the results obtained on the initial set of responses. Due to the, albeit small, differences between the two questionnaires, and to highlight if there is any significant change in the responses over time, the main part of this document (Sections 4 and 5) contains the results obtained on the initial set of interviews, while the updates are reported in the Appendix (Section 9.2).

At any rate, the existing literature was also taken into account when we designed our questionnaire and information extracted from the literature is also summarized in our report.

2 OSS: Issues with the Current Situation

There is a considerable amount of research that is currently being carried out about OSS topics, since many OSS aspects and properties are still not fully understood. In this document, we focus our attention on OSS trustworthiness in the context of industrial environments.

First of all, we need a widely accepted definition of trustworthiness in software systems. One of the first widely accepted and still used definitions is: *the trustworthiness of software is defined as the degree of confidence that exists that the software meets a set of requirements* [1].

The definition is quite broad and, even worse, it is highly subjective. Actually, two levels of subjectivity can be identified:

1. The individuals that evaluate the software: individuals have different confidence parameters and measures.
2. The set of requirements to be satisfied varies depending on how the software system will be used.

The high subjectivity of the trustworthiness concept for software systems should hardly be a surprise. In other engineering sectors, quality and trust are considered to be subjective too, still very important [3]. For a deeper analysis of the possible definitions of trust and trustworthiness in software found in the literature see section 6.1.

The core task to assess software systems' trustworthiness is not to find a general and ubiquitous set of characteristics and parameters to evaluate, but to search, find, and apply a trust evaluation process, tailored to the requirements we want a software system to fulfill.

Despite the subjectivity of trustworthiness, it is a commonly agreed opinion that it is a characteristic that encompasses the reliability, security, and safety of a software system [5]. Trustworthiness is related to fault-tolerance and stability, and some characteristics that a software system must own to be considered trustworthy are [5]: it does not crash at minor flaws; it shuts down in an orderly way in the face of major trauma; it does what it is supposed to do and can repeat that action time after time (producing the same kind of output from the same kind of input). The US National Institute of Standards and Technology (NIST) defines trustworthy software systems as "software that can and must be trusted to work dependably in some critical function, and failure to do so may have catastrophic results, such as serious injury, loss of life or property, business failure or breach of security". However, this is just a minimal set of characteristics, and others need to be considered.

Some authors believe that development and product lifecycle processes play such a major role to become the central aspects to be considered: "A full lifecycle approach to software development is the only way to achieve software trustworthiness" [6].

There also exist methodologies that focus specifically on the trustworthiness of software systems, but these are not widely accepted nor widely adopted. One of these is the Trusted Software Methodology (TSM) [1], which touches a large number of software development practices, characteristics and methodologies, such as: access control, configuration management, documentation, design, development, formal methods, planning, requirements management, security, tools, validation and verification, etc.

Whether OSS is more or less trustworthy when compared to similar proprietary systems is actually still a matter of hot debates and controversial opinions. Even though some believe that OSS is intrinsically at least as trustworthy as proprietary software [2], there are opinions pointing to the opposite ends of the spectrum: from OSS enthusiasts (as in [8]) to much more cautious and skeptical viewpoints (as in [7]).

It is true that there are specific topics where openness is always considered an advantage: for example, free access to source code gives high control on software, in other words it is possible to examine the internal details of the software used. However, this reasoning is quite difficult to extend and generalize to broader domains and more general cases.

A conclusive argument on whether OSS is more or less trustworthy when compared to similar proprietary systems is not available at the current moment. Our opinion is that such a conclusive argument is not to be found any time soon nor is of any practical interest. On the contrary, we believe that trustworthiness of OSS compared to trustworthiness of proprietary systems has to be examined case by case, product by product. Thus, we take a privileged viewpoint -that of the external observer- and our role is to provide means to carry out a dispassionate analysis.

The mentioned facts and open questions constitute the main motivation of our survey. The survey is needed to understand the *confidence parameters of trustworthiness*, the *roles of the involved individuals*, the *requirements* (or, more generally, the *problem domains*) that must be satisfied, and the *relationships* between these three aspects.

It is fundamental to understand what kind of OSS is currently used, and by whom, to correlate the three aspects mentioned above. It is also valuable to understand why a product is preferred over a different product and what makes an OSS solution more appealing to a closed one (or *vice versa*). Our study provides the foundation for answering these questions, even though several of these answers are not given here, but are postponed to work package WP5.3.

The primary and final goal of the A5 activity is to find ways to improve OSS trustworthiness in order to induce more organizations to adopt OSS for more purposes than they have so far. The current study is the first analysis of the current situation (interviews) and of the state of the art (literature reviews) in OSS trustworthiness. As such, it provides input to the workpackages that follow it in activity A5, which will address the primary goal of the activity.

3 The Questionnaire

The interviews have been carried out by using a questionnaire, whose sections and questions address the needs of both A5 (Trustworthy Product) and A6 (Trustworthy Process) activities of the QualiPSo project. The reasons behind this choice are:

- the two aspects are clearly related to each other;
- software professionals are precious resources whose time is very valuable to their companies: having one questionnaire maximized the chances of getting responses from industrial players.

The complete description of the current version of the QualiPSo A5-A6 Questionnaire is in the Appendices (specifically, in Section 9.2).

3.1 Rationale

With our questionnaire, we attempt to obtain answers to the questions detailed in Section 2 and clarify some of the problems mentioned on evaluating OSS trustworthiness. In addition, we try to understand how OSS is perceived by people from different ICT companies and with different roles. Over time, we would like to find out more about the usage patterns and the opinions on OSS matters of people working in EU companies, depending on the people's roles and the enterprise characteristics.

The questionnaire was also developed keeping into account the actual literature on OSS products trustworthiness. Specifically, the various documents and reports analyzed thoroughly in Chapter 6 were used as an input and hint to shape the questionnaire.

The questionnaire we developed is a general-purpose one: it can be used if OSS is used as is or is developed/modified; it is applicable to companies of any size; it targets any role (from the inexperienced developer to upper management levels); and it is not biased by a specific application domain.

The questions in the questionnaire can be mainly classified in three different categories:

1. *Organization, project, and role*. These questions are needed to profile the interviewed person, the company he or she works for, the project(s) he or she participates in.
2. *Actual problems, actual trustworthiness evaluation processes, and factors*. These questions are needed to identify the main factors considered when evaluating whether to adopt an OSS product.
3. *Wishes*. These questions are needed to understand what should be available but is not, and what indicators should be provided for an OSS project to help its adoption.

In several questions, we asked the interviewees to provide an indication of the importance they give to each factor when they adopt OSS products. This importance was measured on a 0 to 10 scale, with value 0 meaning "not important at all" and value 10 meaning "of fundamental importance." The idea was not to actually attach absolutely precise meanings to these numbers, but to provide interviewees with a way to give us their idea of the relative importance among these factors.

3.2 Structure

Here, we concisely describe the overall structure of the questionnaire. It is important to underline that most sections of the questionnaire contained several closed-answer questions and a few open-answer questions, in which the interviewees could provide additional pieces of information, beyond those we had figured could be relevant. The idea was therefore to first guide the interviewees and obtain information about goals and factors that are commonly believed of general interest. These closed-answer questions also allowed the establishment of an effective communication channel between the interviewees and the interviewers, which facilitated the exchange of information. The open-answer questions often resulted in quite interesting and unexpected answers, which shed additional light on the trustworthiness goals and factors of the software industry on OSS and their context.

The questionnaire is divided into several sections. Here, we describe the sections related to Activity A5 (Trustworthy Product).

3.2.1 Personal Information

This information is collected from the interviewee with the following purposes:

- profile the interviewee;
- profile the interviewee's organization;
- profile the organizational unit the interviewee belongs to.

The personal information collected consists of name, email address, role, organizational unit, and education. The organization information collected consists of the organization type, the number of employees, and the application domains of interest. Some of this information is obviously private and is collected for profiling reasons only. It was made clear to the interviewees that this information would not be disclosed at all, and that, in the presentation of results, all information would be disclosed in aggregated form, so as to make it impossible to identify single respondents or single companies and their answers.

3.2.2 Role of the Organization in Relation to OSS

This information is collected to understand the specific use of OSS in an organization: we want to understand whether OSS products are used, integrated, expanded, etc.

The following (non mutually exclusive) categories are used in the questionnaire, to determine the role of the organization in relation to OSS products:

- OSS products are used to support software development;
- OSS products are used as part of other products;

- OSS products are customized and/or configured;
- OSS products are used to support the internal process;
- OSS products are used to provide services to the outside world;
- OSS is the development platform;
- OSS is the target and/or usage platform.

3.2.3 Economics: Economic Issues When Choosing OSS

This information is collected to understand the main economic drivers behind the choice of a specific OSS product over other OSS products or closed source software. We focus primarily on Return On Investment (ROI) and Total Cost of Ownership (TCO).

3.2.4 License: License Issues When Choosing OSS

This information is collected to have information on the most widely used licenses, the possible recurring problems in using available licenses, and, in general, the characteristics that a good or ideal license should have (what a license should permit, what it should deny, etc.).

This information can also be useful to understand whether there is the need for a specific license to meet the requirements of EU organizations, or, on the contrary, whether a subset of currently available licenses covers the needs of EU organizations.

It is to be noted that the intent of these questions is not to clarify or better understand legal issues concerning licenses. The intent is to understand the wishes and the perceptions of the interviewees regarding licenses related to software trustworthiness and the selection process of OSS products.

The license characteristics we considered relevant, and that are listed in the questionnaire, are the following:

- Hackers like to accept code under the license.
- License permits to combine the considered OSS with proprietary software and redistribute.
- License permits to combine the considered OSS with GPL licensed software and redistribute.
- License permits to redistribute binaries without source.
- License is applicable to anyone who receives the software system, without the need for any additional agreements.
- License permits to combine the considered OSS with software, with any agreement and license, and redistribute.
- License permits to redistribute the software in any form.
- License permits to redistribute the software charging money for it.
- License permits to access to the source code.

- License permits to modify the software.
- License permits to redistribute modified versions of the software.

3.2.5 Development: OSS Development Process

We collected information to understand the process used to select a specific OSS product, even in cases where the process is completely informal. In addition, we want to determine the main development process factors when choosing an OSS product and the available attributes that are taken in consideration. We also seek to know what attributes that are currently not available would supplement the selection process and make it more effective.

The attributes usually available that can be taken in consideration when choosing OSS that we valued important are the following:

- type of licenses used;
- the availability of tools for developing and/or modifying and/or customizing OSS products;
- the availability of best practices for the specific OSS products;
- the availability of technical documentation and/or of a user manual;
- environmental issues (i.e., the issues related to any constituent of the environment, such as the platform used, the usable personnel, the available hardware, etc.);
- the availability of training, guidelines, etc.;
- the mid- or long- term existence of a user community;
- the mid- or long- term existence of a maintainer organization and/or of a sponsor;
- the short-term support for problems resolution, corrections of bugs, etc.;
- the reputation of the OSS provider;
- the programming language uniformity, to understand if an application that uses only one language or a limited number of languages, is preferable over applications built using a large set of languages;
- the existence of a sufficiently large community of users that can witness the OSS product quality;
- the existence of benchmarks and/or test suites that can witness the OSS product quality.

3.2.6 Quality: Product Quality Issues When Choosing OSS

This information is collected to understand the product quality attributes that OSS users take into account when selecting OSS products (OSS user has the broad meaning we defined in Chapter 1).

We investigated two kinds of quality attributes:

- *external attributes:* attributes related to the various *OSS users*: requirements satisfaction, reliability, performance, usability, maintainability, portability, interoperability, human interface language, localization, self-containedness (the product does not need other products to work); the ISO 9126 qualities [41] were chosen in the questionnaire as a primary reference for external product quality attributes;
- *internal attributes:* attributes related to the intrinsic structure of the software code, design, requirements, etc.: size, complexity, modularity, standard architecture used, patterns used, standards compliance.

Thus, external attributes are *OSS user* related attributes, that is, attributes that can be perceived directly by *OSS users*. On the other hand, internal attributes are attributes related directly to source code or software development artifacts.

3.2.7 Customer: Customer Requirements

By customer, we mean the person that has requested a service, a system, a library, a tool, etc. Sometimes the customer coincides with the OSS user, but when we precisely reference a customer, we intend to stress out a purchaser role. Often the customer will also directly define the requirements of the product to be purchased. It is to be noted that also a development team could have a customer role, when, for example, requesting a development tool or a specific library.

The information is collected to understand how influential *customer* requirements are when choosing an OSS product. The requirements considered in the questionnaire are:

- customer satisfaction;
- interoperability constraints, since the OSS product has to be integrated with other systems;
- law constraints; that is, the OSS product has to meet the requisites of one or more specific laws. A well known example is the USA law restrictions on exporting software products that contains cryptographic algorithms.
- standard constraints (the software system has to fulfill one or more standards).

3.2.8 Factors

The resulting list of factors, extracted from the questionnaire, is shown in Table 3-1.

Actually, two more factors, *reusability* and *security*, were added, because they were often[2] mentioned by interviewees as factors to be considered when selecting an *OSS product*. The column 'Section', in Table 3-1, represents the section of the Questionnaire where the factor is to be found.

[2] In the questionnaires collected, many new factors that should be considered when selecting an OSS product are suggested from the interviewees. Nevertheless the two newly added factors (*reusability* and *security*) are the only ones that have a meaningful statistical relevance.

The added factors *reusability* and *security* appear in a few observations, which are enough to show the interest in these two factors, but not to obtain entirely significant results when we compared their relative importance to the importance of the other factors, since a sufficient number of statistically significant relationships is needed to reliably rank the relative importance of factors. Hence, the factors appear in the statistical analysis reported in Appendix 9.1 for completeness, but they were removed from the results presented in Chapter 5.

Table 3-1 Questionnaire factors (section column represent the section of the questionnaire where the factor is to be found)

section	***factor name***	***mnemonic factor name***
Economic	ROI	ROI
Economic	TCO	TCO
Development	types of licenses used	type of licenses
Development	availability of tools for developing modifying customizing OSS products	tools
Development	availability of best practices on the specific OSS products	best practices
Development	availability of technical documentation / user manual	documentation
Development	environmental issues	environment
Development	availability of training, guidelines, etc.	training / guidelines
Development	mid- / long- term existence of a user community	user community
Development	mid- / long- term existence of a maintainer organization / sponsor	maintainer organization
Development	short-term support	short term support
Development	reputation of the OSS vendor	reputation of vendor
Development	distribution channel	distribution channel
Development	programming language uniformity	language uniformity
Development	existence of a sufficiently large community of users that can witness its quality	user community that witness quality
Development	existence of benchmarks / test suites that witness for the quality of OSS	benchmarks / test suites
Quality	degree to which an OSS product satisfies / covers functional requirements	functional requirements
Quality	external quality reliability	eq reliability
Quality	external quality performance	eq performance
Quality	external quality usability	eq usability
Quality	external quality maintainability	eq maintainability
Quality	external quality portability	eq portability
Quality	*external quality reusability*	eq reusability
Quality	internal quality size	iq size
Quality	internal quality complexity	iq complexity
Quality	internal quality modularity	iq modularity
Quality	internal quality standard architecture	iq standard architecture
Quality	internal quality patterns	iq patterns
Quality	*internal quality security*	iq security
Quality	standard compliance	standard compliance
Quality	self containedness	self containedness
Quality	interoperability	interoperability
Quality	human interface language / localization	localization
Customer	customer satisfaction	customer satisfaction
Customer	interoperability issues	interoperability issues
Customer	law conformance	law
Customer	standard imposed	standard imposed

4 THE INTERVIEWS

As already explained in the Introduction, this section and Section 5 contain the results obtained on the initial set of 103 interviews. Section 9.2 in the Appendix part of this document contains an updated version of the results, obtained on he total of 151 interviews we have collected. As noted in the Introduction, we have kept them separate because they combine results obtained two slightly different version of the questionnaire, and to check if there are any changes in the responses obtained over time. At any rate, Section 9.2 in the Appendix also shows that the vast majority of the initial results was confirmed in the combined set of interviews.

4.1 Conducting interviews

In most of the cases, data collection has been carried out with synchronous communication with the interviewees, that is, by in person or phone interviews. We believe this is the most effective way to elicit information and establish an effective communication channel with the interviewees. We wanted to collect information that was structured by means of closed-answer questions and additional information by talking with the interviewee.

We also carried out interviews by email, giving feedback and advice in an asynchronous way. The results seem to be fairly aligned and coherent with the direct interviews, but of poorer quality, as the answers are often far less weighed up. Also, we obtained far fewer details on open questions (that is, on questions where a complete and exhaustive answer is needed).

When the first analysis of the data collected was carried out, the differences between the questionnaire obtained in a synchronous way and the questionnaires obtained in an asynchronous way become clear. Hence, we decided to continue with interviews only in person or by telephone. Most of the questionnaires obtained in an asynchronous way have been later integrated with explanations and notes obtained in a synchronous way, getting in touch again with the interviewee, whenever possible.

All the interviews we carried out were individual ones, usually with one interviewee at a time, since we believed that it is important that the interviewees provide their own viewpoint without any sort of conscious or even unconscious interference due to the presence of other people, especially if belonging to the same organization.

While conducting the interviews, the feedback received from the first interviewees allowed us to revise and improve the questionnaire, until it soon reached its current version.

4.1.1 Automated data collection

An online version of the questionnaire is available on our intranet website to allow for automated data collection, which facilitates data analysis.

The tool to gather the questionnaire responses was developed based on a LAMP (Linux, Apache, MySQL, PHP) stack solution. The Questionnaire was developed using the PHP library LimeSurvey (formerly known as "PHPSurveyor") (http://www.limesurvey.org).

The online questionnaire is not publicly available since experience showed that interviews are most effective in person or by telephone. Thus, the only apparent benefit of having an online questionnaire would be to increase the total number of interviews. However, since the interviewees cannot be properly identified, nor the answers to the questionnaire can be validated, we chose quality over quantity and did not use the online questionnaire in the initially intended way.

However, online data collection, even if not directly used, has been useful to us as a way to collect and organize answers in a database.

4.2 The Sample

We have collected 103 interviews. The nationalities of the interviewees comprise several countries of the QualiPSo participants (Italy, Germany, France, Spain, Poland, Brazil, British, China) and two others (United Kingdom and USA).

The current sample is heterogeneous, not only considering the nationalities of the interviewees, but also considering:

- the role of the interviewee
- the type of the organization of the interviewee
- how OSS is used by the interviewee.

It must be noticed that the population of interviewees was not determined in advance. We did not define in advance how many interviewees of a specific kind were to be included in the set. A planned sample set would have allowed a more controlled result analysis, but it would also have limited the possibility to add interviewees to the set in an unanticipated manner. Thus, we decided to keep the set of interviewees open. If required, it would be possible to carry out additional data analysis based on some desired distribution in our sample.

In any case, the sample obtained is suitable to achieve the goal to get a comprehensive understanding of the factors that influence the selection of an OSS product and its trustworthiness.

Before proceeding to the extraction of results, the sample was analyzed. It was found that a subset of the interviewees were potentially problematic for our purposes. Specifically, those interviewees that belong to public administration organizations and that do not hold management roles (these include regular non management jobs as well as researchers) showed little interest in economic issues or a poor understanding of the rationale behind supplying OSS products. More importantly, they did not appear to have an important role when choosing OSS products.

Therefore, we decided to derive two (sub) sets, to be both, separately, analyzed:

- A "clean" set, which we call the *Main set* , in which the "problematic" interviews do not appear (78 interviews).

- The entire set, which we call the *Whole set*, which includes all the interviewees (103 interviews).

The *Main set* was used for the main analysis, whose results are reported in Chapter 5. The *Whole set* has been studied and analyzed as well. The main differences between the results obtained with the two sets are shown in the statistical details reported in the Appendices (Section 9.1).

4.3 Distribution of the interviewees

4.3.1 Roles

Possible answers:

- Upper management (Yes, No)
- Project manager (Yes, No)
- Developer (Yes, No)
- OSS expert (Yes, No)

The answers are not mutually exclusive, i.e., an interviewee could provide multiple "Yes" answers for these questions (for example, an interviewee could be both a Developer and a Project Manager).

The roles of the interviewees are fairly equally distributed among Upper managers, Project managers and Developers (see Figure 4-1). OSS experts are clearly underrepresented.

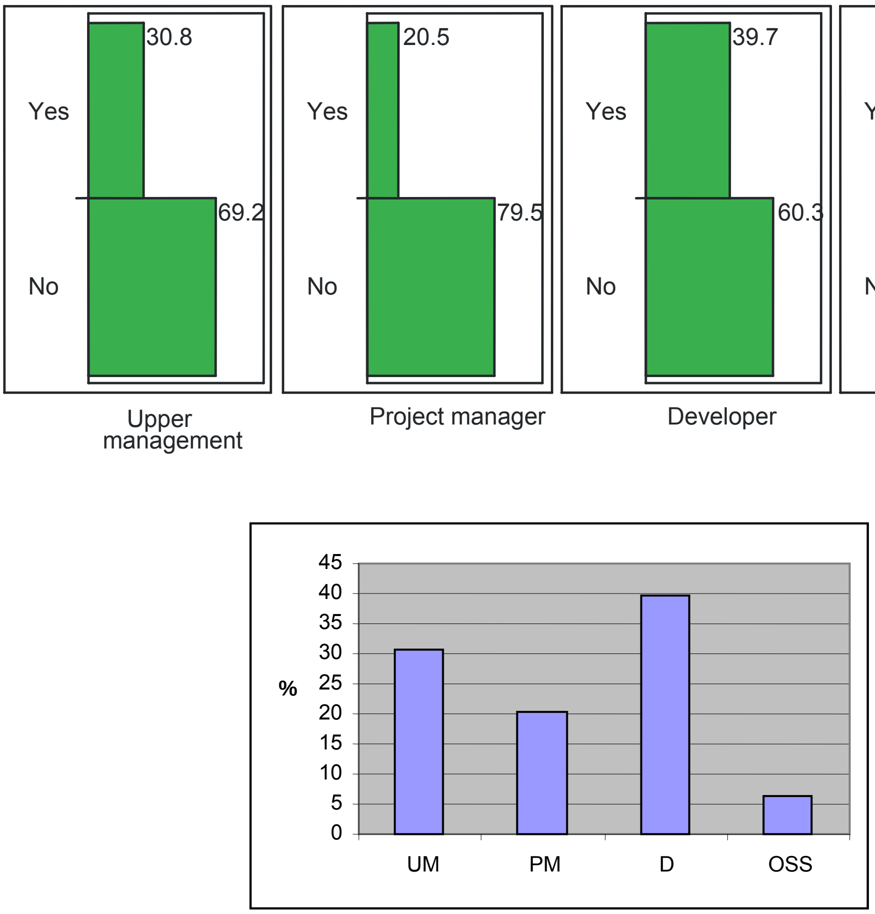


Figure 4-1: Roles answers (percent values are shown, the answers are not mutually exclusive).

In addition, the interviewees seldom played multiple roles, and we found only 12 cases of overlapping roles.

4.3.2 Education

The answers connected to the education of the interviewees (again, the Yes or No answers are not mutually exclusive) are distributed as shown in Figure 4-2.

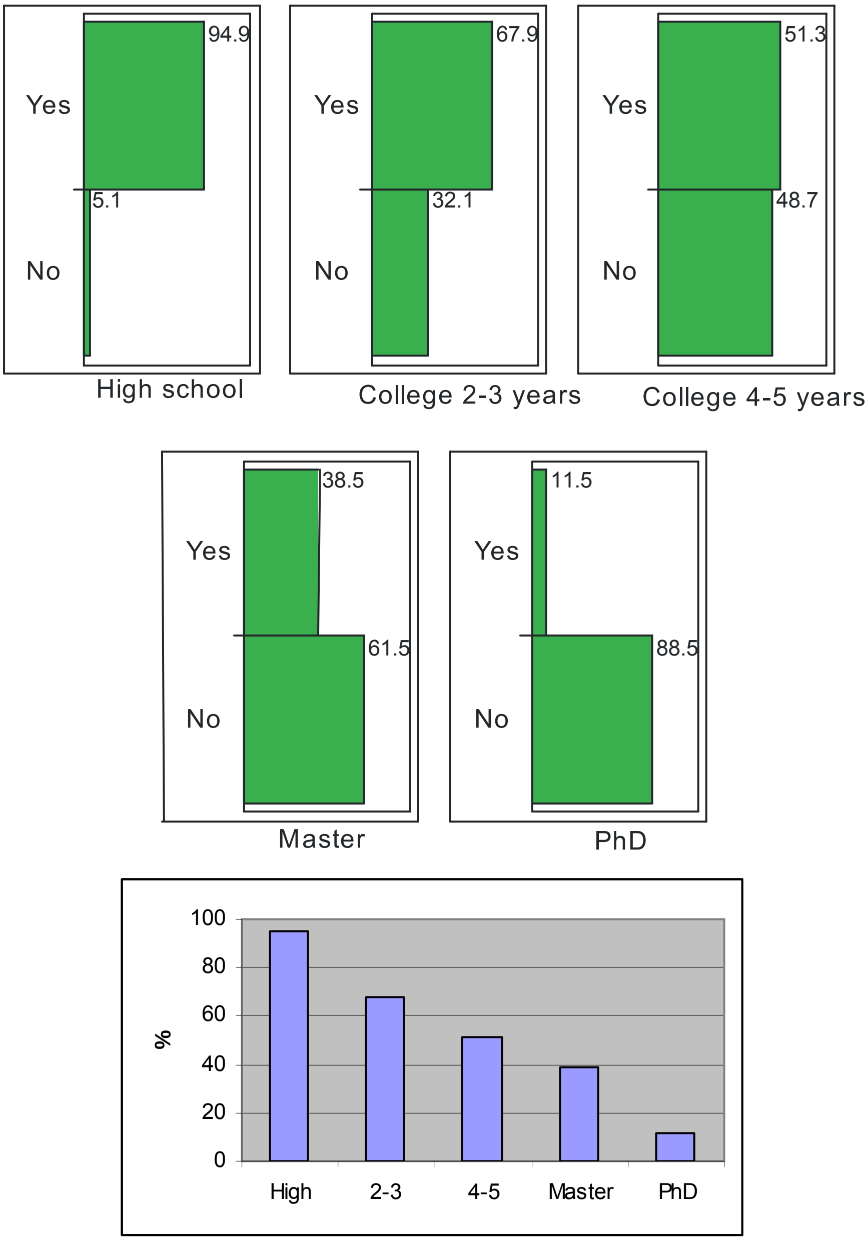


Figure 4-2: Education answers (percent values are shown, the answers are not mutually exclusive).

Notice that the interviewee can provide multiple "Yes" answers for the education degrees questions. Multiple "Yes" answers imply that each respondent can have more than one degree of education. For example, a *PhD* degree also implies *High school*, *College 2-3 Years* and *College 4-5 years* degrees. This explains the high percentage of High school and College 2-3 years (both are implied by *PhD* and *Master* degrees).

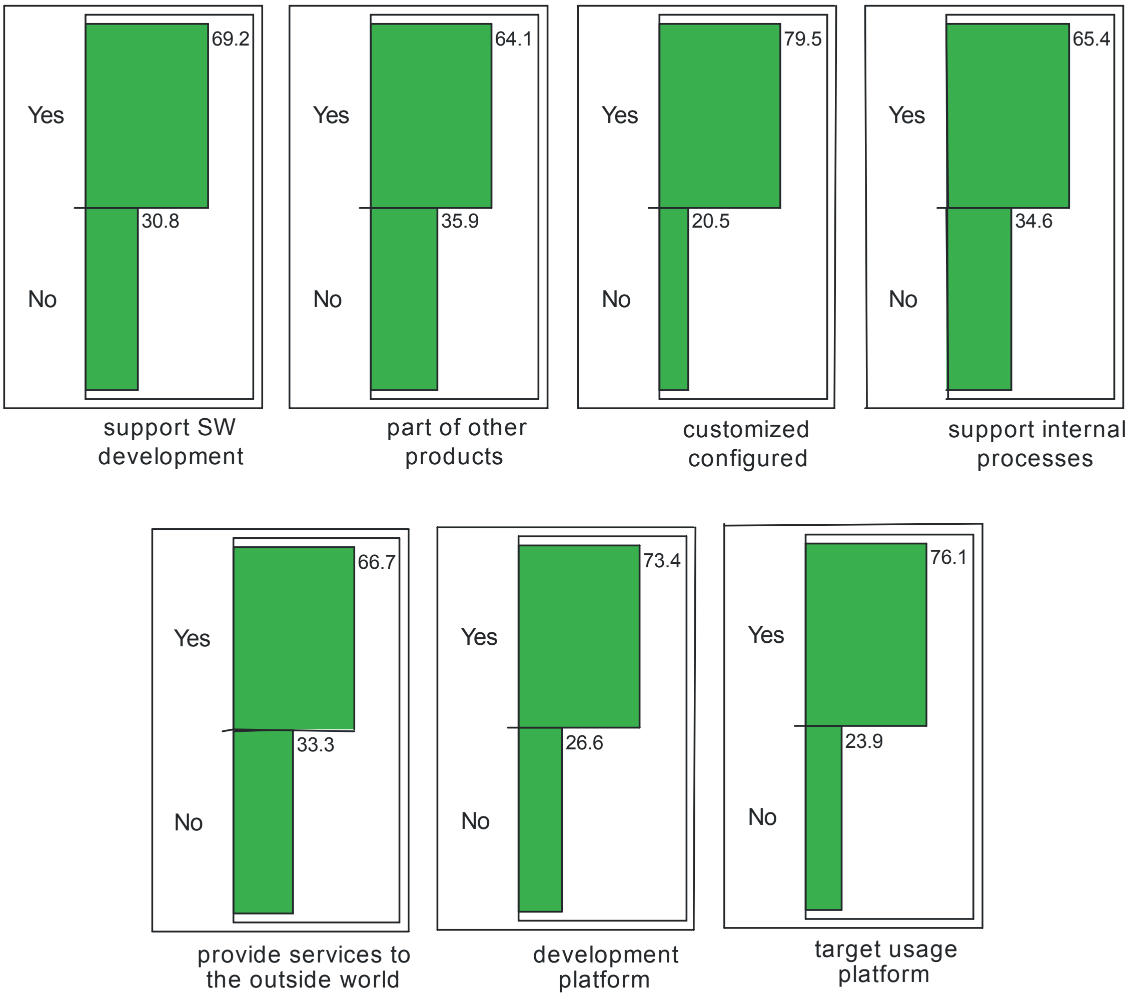
The interviewees that declared a school level below *High school* (no degrees at all) are only the 5.1%; this corresponds to the percent of *High school* answers to "No".

4.3.3 Roles of OSS in the Organization

Possible answers (answers are not mutually exclusive):

- OSS products are used to support SW development (Yes, No)
- OSS products are used as part of other products (Yes, No)
- OSS products are customized / configured (Yes, No)
- OSS products are used to support internal processes (Yes, No)
- OSS products are used to provide services to the outside world (Yes, No)
- Is OSS the development platform? (Yes, No)
- Is OSS the target / usage platform? (Yes, No)

All possible roles of OSS in the organizations are well represented, as shown in Figure 4-3.



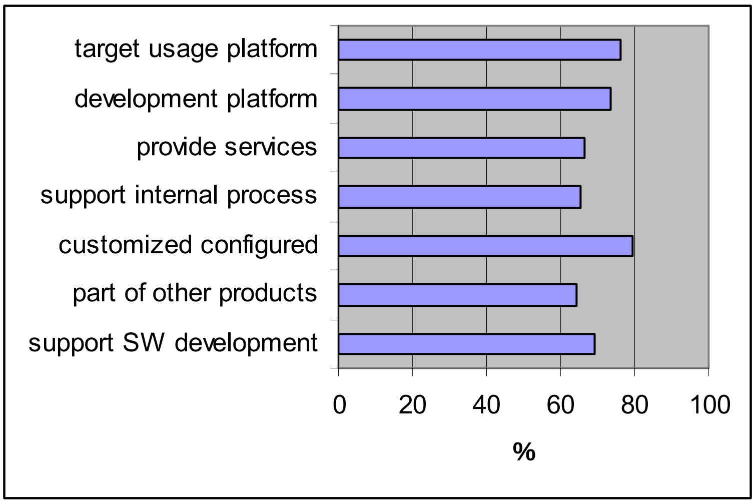


Figure 4-3: Roles of OSS in the Organization answers (percent values are shown, the answers are not mutually exclusive, questions are shown in an abbreviated form).

Many of the Organization roles show a high overlapping rate, since each organization holds more than one role at the same time, as could be expected. Only a small percent (3.8%) of the interviewees answered "No" to all 7 questions, meaning that OSS has no roles at all in the organization of the interviewee.

4.3.4 Type of Organization

The main part of interviewees is employed in Private organizations, while a smaller but significant part is employed in Public organizations and No profit organizations, see Figure 4-4.

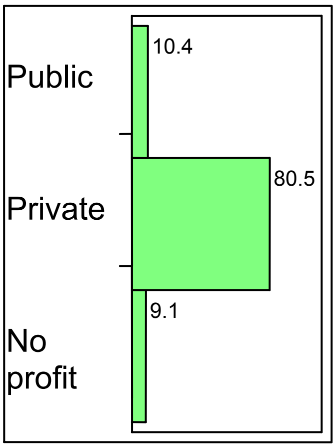


Figure 4-4: Type of Organization answers (percent values are shown).

4.4 Associations between Interviewees' Characteristics

Here, and in all the following statistical tests, we used a threshold of 0.05 for the statistical significance of the results, as is usually done in Empirical Software Engineering studies. This means that when we say that we have found, say, an "association" between two factors, there is actually a sufficient statistical evidence to support the claim that the two factors are related.

We analyzed Roles, Organizational roles and Organization type for associations, using Fisher's exact Test [9], and we found no unusual or unexpected associations.

Expected association between roles were found, thus confirming the coherence of the sample; for instance, an Upper Management role usually negates a Developer role and *vice versa*.

We found many associations between the roles of OSS in the organization (see Table 4-1), confirming the intuition that all these roles are tightly connected. It is to be noted that the associations between Organization roles are always concordant (Yes-Yes, No-No): for example, the first association shows that an organization that uses OSS products to support software development probably also uses OSS products as part of other products.

Table 4-1: Organization roles associations (all the associations are concordant).

factor	***factor***
org role support sw development	org role part of products
org role support sw development	org role support internal processes
org role support sw development	org role development platform
org role part of products	org role customized / configured
org role part of products	org role support internal processes
org role part of products	org role provide services
org role customized / configured	org role provide services
org role support internal processes	org role provide services
org role support internal processes	org role development platform
org role support internal processes	org role target platform
org role provide services	org role target platform
org role development platform	org role target platform

Other associations found (using Pearson's χ^2 Test [12]) involve the Type of organization: Public Administration organizations usually have different distributions when compared to Private organizations; in Public Administrations the level of education is higher than that found in Private organizations (see Figure 4-5).

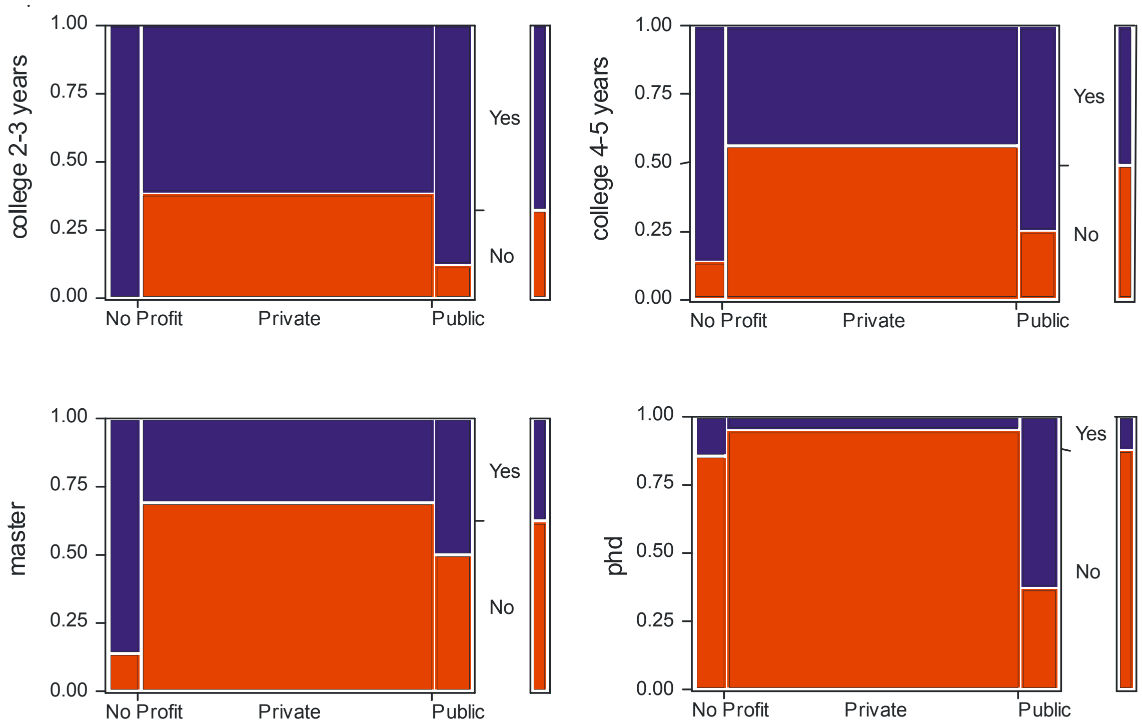


Figure 4-5: Education degrees depending on Type of organization.

5 GOAL AND FACTOR QUALITATIVE ANALYSIS

Here, we provide a concise analysis of the responses we have obtained by means of the questionnaire, with the insights gained by the statistical analysis we performed (details can be found in Appendix 9.1). Recall that we asked the interviewees to estimate the relative importance of factors on a 0 (totally unimportant) to 10 (absolutely important) scale.

The summary of the results is shown in Table 5-1, in which the factors, which were introduced in section 3.2.8 are ordered in decreasing order of importance. The statistical analysis has actually allowed us to partition the factors in 8 separate groups, and has provided evidence for the existence of an ordering between factors belonging to different groups. For instance, factor *customer satisfaction* (group 7) is believed to be more important than *iq modularity* (group 6). No ordering can be established among the factors belonging to the same group. For instance, we do not have supporting evidence to say that *eq reliability* is more important than *eq maintainability* or vice versa, since both are in group 7.

The existence of groups with factors having similar importance was an expected outcome of our data analysis, since we have a number of factors to order and the size of our sample is not too large, even taking into account the entire data set. In terms of the clustering of the results, we notice that there are 2 relatively larger groups (the ones whose ranking is 3 and 5), 5 other smaller groups, and 1 singleton group with the highest ranking.

Considering the ordering relations found, we consider the groups with an increasing level of importance:

1. negligible importance
2. very low importance
3. low importance
4. medium importance
5. fairly high importance
6. high importance
7. very high importance
8. fundamental importance

Table 5-1: Factors groups (*Main set*), ordered by group

section	*factor name*	*mnemonic factor name*	*group*	*importance*
Quality	the degree to which an OSS product satisfies / covers functional requirements	functional requirements	8	fundamental
Customer	customer satisfaction	customer satisfaction	7	very high
Customer	interoperability issues	interoperability issues	7	very high
Quality	external quality maintainability	eq maintainability	7	very high
Quality	external quality reliability	eq reliability	7	very high
Quality	interoperability	interoperability	7	very high
Development	the availability of technical documentation / user manual	documentation	6	high
Development	the mid / long term existence of a user community	user community	6	high
Quality	internal quality modularity	iq modularity	6	high
Quality	internal quality standard architecture	iq standard architecture	6	high
Quality	standard compliance	standard compliance	6	high
Customer	law conformance	law	5	fairly high
Development	environmental issues	environment	5	fairly high
Development	the availability of tools for developing modifying customizing OSS products	tools	5	fairly high
Development	the existence of a sufficiently large community of users that can witness its quality	user community that witness quality	5	fairly high
Development	the short term support	short term support	5	fairly high
Development	the type of licenses used	type of licenses	5	fairly high
Economic	ROI	ROI	5	fairly high
Quality	external quality performance	eq performance	5	fairly high
Quality	external quality usability	eq usability	5	fairly high
Development	the reputation of the OSS vendor	reputation of vendor	4	medium
Quality	external quality portability	eq portability	4	medium
Customer	standard imposed	standard imposed	3	low
Development	the availability of best practices on the specific OSS products	best practices	3	low
Development	the existence of benchmarks / test suites that witness for the quality of OSS	benchmarks / test suites	3	low
Development	the programming language uniformity	language uniformity	3	low
Economic	TCO	TCO	3	low
Quality	human interface language / localization	localization	3	low
Quality	internal quality complexity	iq complexity	3	low
Quality	internal quality patterns	iq patterns	3	low
Quality	self containedness	self containedness	3	low
Development	the availabiliity of training, guidelines, ecc.	training / guidelines	2	very low
Development	the mid / long term existence of a maintainer organization / sponsor	maintainer organization	2	very low
Development	the distribution channel	distribution channel	1	negligible
Quality	internal quality size	size	1	negligible

The factors reported in Table 5-1 are analyzed in details in the following sections. The sections are structured following the structure of the Questionnaire (see section 3.2): Economics, License, Development, Quality and

Customer sections. The only exception is on the section dedicated to the Selection Process (section 5.1).

Each section, except the License section (Section 5.3) and the Selection Process section (Section 5.1), is organized as follows.

1. Quantitative and qualitative analysis of the results for all the factors of the corresponding questionnaire section. All the factors are summarized in a table that shows their importance groups and the mean values[3] of the responses we obtained for each factor.
2. List of the most frequent responses to open questions. We report these answers because they can help have a better understanding of the issues that are relevant when selecting OSS.

5.1 OSS Selection Process

The majority of respondents answered that they do not use a formal OSS selection process; but, when they were asked further, they admitted that they actually do use an informal selection process, roughly followed in the respondents' organizational unit; the selection process statistics are summarized in Figure 5-1.

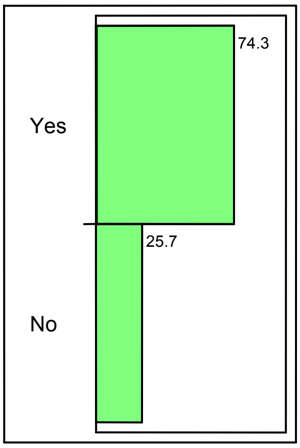


Figure 5-1: Existence and usage of an OSS Selection Process (percent values are shown)

In same cases, interviewees declared that legal aspects are taken into account in the selection process, even directly involving the legal department in the selection process.

None of the respondents mentioned the use of the existing OSS product evaluation methods that are available in the literature [29][30][31][32][34]. This

[3] We include mean values because they provide an expressive idea of the average sentiment of our respondents. However, due to the kind of data we collected (nominal or ordinal data) our statistical analyses are based on so-called nonparametric tests (as recommended by the statistical literature), which do not make use of means. We used these nonparametric tests to cluster our factors in importance groups.

shows that there is a gap to be bridged between these methods and the practice, even though some of the methods originated in software companies.

5.2 Economics: Economic Issues When Choosing OSS

In general, both Return On Investment (*ROI*) and Total Cost of Ownership (*TCO*) were expected to be considered very important, but the results do not support this intuition, even relegating *TCO* in group 3 (low importance) of the *Whole set* groups.

These unexpected results could be partly explained considering that in the *Main set* we can find several developers that seldom address directly economic factors. This intuition is confirmed when examining the *Whole set*. Table 5-2 illustrates the importance given to TCO and ROI by the Whole set of interviewees as opposed to the Main set. In Table 5-2 fact, columns "main mean" and "whole mean" in Table 5-2 denote the mean values of the score given to the factor by the respondents in *Main set* and *Whole set*, respectively. The economic factors in the *Whole set* are considered of a lower importance than in the *Main set*.

Table 5-2: Economics factors: *Whole set* and *Main set*, group and mean

section	*factor*	*main group*	*main mean*	*whole group*	*whole mean*
Economic	ROI	5	6.362	3	5.722
Economic	TCO	3	6.081	2	5.633

Another rather unexpected finding is that *ROI* has higher importance than *TCO* (in both sets), since when comparing OSS products to closed source, proprietary products, *TCO* is usually considered a more relevant and direct indicator than *ROI*.

5.2.1 Answers to open questions

Other economic related factors and issues have been mentioned as important by the respondents as a part of the open questions. Here, we report a summary of the issues collected.

- **Ethics**. OSS experts and OSS supporters support ethic values instead of direct economical profits.
  - **Social cost**. Social cost is considered as important as direct cost; this factor can be related to the more general ethics factor.
- **Development time**. Delivery time is held as more important than the total cost of the product.
- **No use**. A very small number of the interviewees' organizations do not use OSS products *a priori.*
- **Closed specifications**. In some organizations, software systems are developed to fulfill closed specifications, which cannot be freely distributed. Hence, the implementation of closed specifications in a software product to be distributed sometimes negates the possibility to use OSS products.

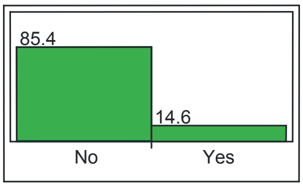
- **Integration cost and effort**. Some products need to be integrated with existing software. Integration cost and effort have been reported to be high if there is the need to integrate proprietary software with OSS.
- **Risk analysis**. Risk analysis is the most important analysis done to evaluate the acquisition and use of a software system.
- **OSS market**. The possibility of becoming the driving force behind some OSS market niche by developing OSS is considered as the one of the driving economic factor.
- **Differentiate from competitors**. OSS software can be a distinguishing factor when compared to competitive products. This factor is going to lose some of its strength once OSS software is more widely adopted.
- **Full control of code**. This is considered an important economic factor, since unwanted economic dependencies can be avoided.
    - **Ability to contribute to evolve and adapt the software**. This factor can be considered a sub factor of full control of code.
    - **Independence from specific vendors and commercial products**. Independence and no vendor lock-ins are very important economic dependencies to be avoided. This factor can be considered a sub factor of full control of code.
- ***ROI***.
    - **Absence of license fees**. This factor stresses the fact that software licenses will be acquired for free, hence increasing the *ROI*.
    - **Try many solutions without spending money**.
- ***TCO***.
    - **Preference to stay with the same OSS product because expertise was acquired, and this reduces the effort**. This factor can be seen as a characteristic of *TCO*.
- **Acquisition**.
    - **Ease of acquisition**. Ease of acquisition, especially for support and assistance services of OSS products, is considered important
    - **Rules for spending money**. In many organizations, spending money to buy software can be a lengthy and complicate process. Since there is usually no money to be spent at the moment of OSS acquisition, OSS is regarded as a faster and easier way to acquire the needed software.

5.3 License: License Issues When Choosing OSS

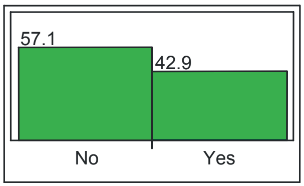
Some interviewees identified a large number of licenses that are used in their organization, while the vast majority only named a few. Most of the interviewees considered licenses and legal issues important when incorporating an external OSS product in their own products: the factors *type of licenses* and the factor *law* are in group 5 (fairly high importance).

Oftentimes, OSS products come with other licenses that are not explicitly mentioned. Clarity in the licenses is a common requirement, since it is often difficult to understand what a license allows or prohibits OSS users to do. The large number of existing licenses further complicate this issue, since some of the licenses appear to be similar, but turn out not to be fully compatible. This is a well known hindrance to the adoption of OSS and the business of OSS users.

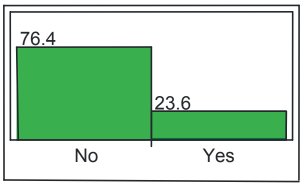
Some generic license questions were asked in the questionnaire, to understand how the licenses are perceived, and what characteristic an ideal license should have. The questions asked were not easily understood, and around the 30% percent of the interviewees did not answer them. In Figure 5-2 the answers are shown, for each characteristic an ideal license should have.



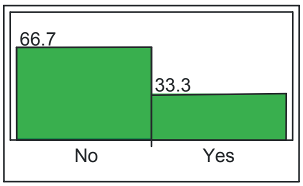
hackers dislike accepting code covered by the license



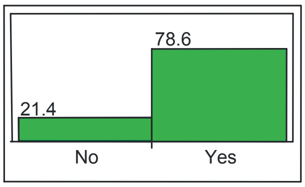
cannot combine with proprietary software and redistribute



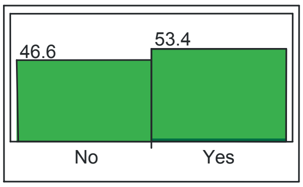
cannot combine with gpl'ed code and redistribute



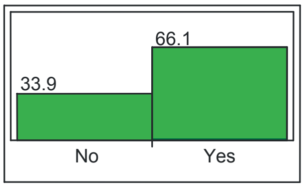
can redistribute binaries without source



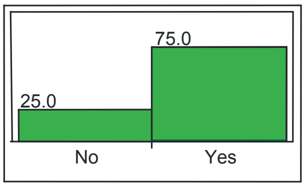
apply to everyone who receives the program



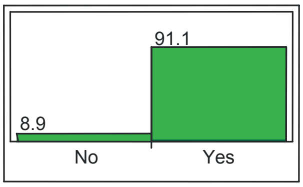
allow distribution with any other software agreements



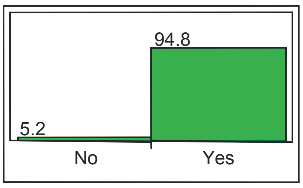
allow distribution in any form



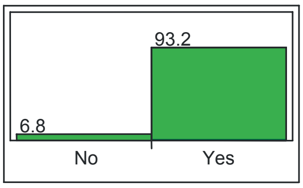
grant to distribute the program themselves including the right to charge money for it



grant the right to distribute modified versions of the program



grant access to the program's source code



grant the right to modify the program

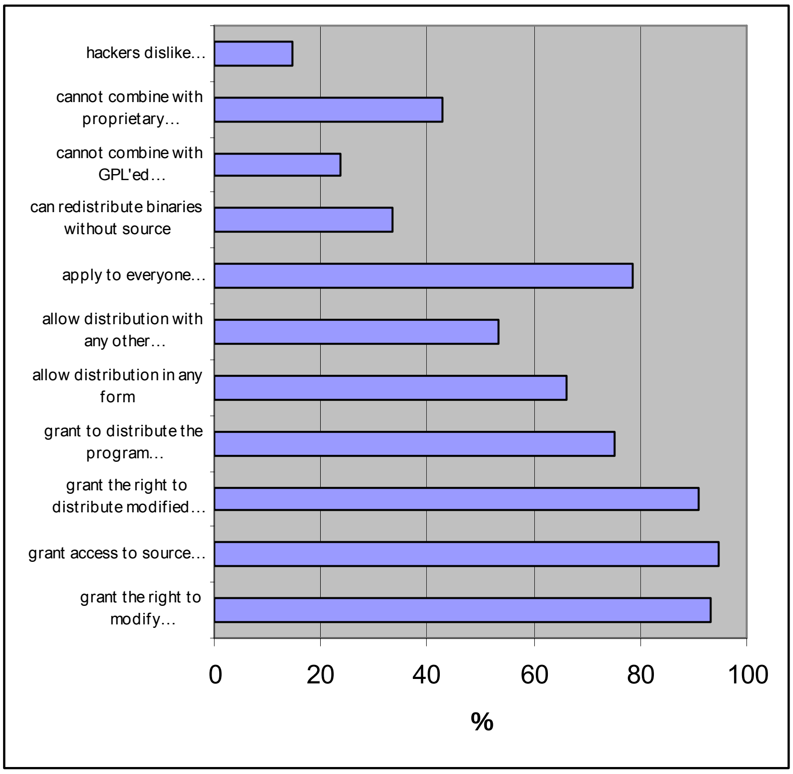


Figure 5-2: License characteristics (percent values are shown)

5.3.1 Answers to open questions

Here are the most important and interesting comments on license-related issues, requests, and factors.

- **Allow everything**. One of the desired licenses allows everything and restrict nothing.
- **GPL**. In a way, GPL is considered as the standard license. In some organizations, every product is requested to use a GPL license or a license compatible with the GPL license.
- **Applicability issues**. An OSS license is requested to be applied to everyone who receives the program, without the need for any additional agreements. This is an issue with the current laws and regulations in many European countries.
- **Multiple licenses**[4]. Some interviewees consider the availability of a multiple license model an important advantage, since it guarantees support and a

[4] Multiple licensing is the practice of distributing identical software under different sets of terms and conditions. This may mean different licenses, or different sets of licenses. Multiple licensing is sometime used to support OSS business models. An OSS product is proposed in at least two licenses, a traditional proprietary license and an OSS license. With the proprietary software license it is possible to actually sell the product with services and technical assistance, while with the OSS license the product is released free (free to modify, free to redistribute, free of charge, etc.) with no additional services or assistance.

sustainable business model for the vendor, while others considers it a hindrance, since it could generate unwanted vendor dependencies[5].

- **License management**. License issues and uncertainty on license compatibilities are perceived as a problem that needs to be solved to increase adoption rate and use of OSS products.

5.4 Development: OSS Development Process

In general, the interviewees showed interest (even though with various degrees) in OSS. In some cases, OSS could not be used because no suitable OSS components were available, or because the available OSS components were not certified, while the applicable regulations mandated that software be certified.

Some interviewees check the quality of an OSS product by testing it thoroughly. At any rate, the factor *benchmarks / test suites* is considered of low importance (the factor lies in group 3).

Table 5-3: Development factors: group and mean

section	*factor*	*group*	*mean*
Development	type of licenses	5	6.803
Development	tools	5	6.569
Development	best practices	3	6.181
Development	documentation	6	7.733
Development	environment	5	6.836
Development	training / guidelines	2	4.880
Development	user community	6	7.284
Development	maintainer organization	2	5.640
Development	short term support	5	6.987
Development	reputation of vendor	4	5.595
Development	distribution channel	1	3.417
Development	language uniformity	3	5.806
Development	user community that witness quality	5	7.054
Development	benchmarks / test suites	3	5.616

Documentation availability is considered an important selection process factor: *documentation* lies in group 6 (high importance).

The environment and the context play significant roles in the OSS selection, and this is confirmed by the factor *environment* that lies in group 5 (fairly high importance).

The analysis indicate that interviewees do pay attention to the vitality of the user community, in terms of its duration and, to a lesser degree, the number of people involved[6]: *user community* lies in group 6 (high importance) while *user*

[5] Multiple license OSS products examples: MySQL (http://www.mysql.com), Qt (http://www.trolltech.com), Sleepycat (http://www.sleepycat.com).

[6] A small number of developers and a small community do not necessarily imply a bad product. This is especially true when dealing with niche applications, which usually have a small developers community as well as a small user community.

community that witnesses quality, *short term support* (the possibility to have bugs fixed in a short period of time) and *tools* lie in group 5 (fairly high importance).

Interviewees are not very interested in the existence of a sponsor organization behind an OSS product: the corresponding factors *reputation of vendor* lie in group 4 (medium importance), and *the mid / long term existence of a maintainer organization / sponsor* lie in group 2 (very low importance). The respondents who are less interested in such an organization are usually willing to carry out the required modifications to the chosen OSS by themselves.

Best practices is not believed to be an important factor, even though this factor is somewhat similar to the documentation factors: this factor lies in group 3 (low importance). Other factors that are not considered important are *language uniformity* and *training / guidelines*: both are considered of low importance, since *language uniformity* lies in group 3 and *training / guidelines* lies in group 2.

5.4.1 Answers to open questions

At any rate, some respondents mentioned that they would like to have the following additional information that is hardly ever available.

- **Rationale**. The rationale behind developing the OSS product. Among other things, the motivations of strategic decisions that led to the development of the OSS should be clear and documented.
    - **Roadmap**. A roadmap could be easier to obtain if there is a driving force, i.e., a sponsor organization behind the OSS product.
    - **Release history**. A release history should be available. Every release should come with documentation, explanation of the motivations and of the design choices made, and the indication of the effort spent.
    - **Expected lifetime**. It is not always clear what the expected lifetime of the software is (in other words, we need an answer to the following question: "Is the community big enough and the product interesting and good enough to sustain the development?"). In the long term, it is important that a given OSS product stays alive so that it can be used in the next generation of products reusing existing knowledge.
- **Development approach visibility**. The development process description should answer questions, among others, like "What activities are done and by whom?", "What tests are executed?", "What is the frequency of delivery?" etc. Since the development process is usually not formalized, the code can be heterogeneous in style and quality and this may complicate the integration with other software and the modification of the source code. Interviewees suggest that a brief description of the coding style, conventions, etc. could be useful to shorten the time needed to get acquainted with certain OSS products.
    - **Public list of active developers**. The purpose would be mainly to facilitate the communication with accountable persons. Public lists should

make it easier to answer questions like "Who can submit a new feature?", "Who can submit a patch?", "How a new feature can be submitted?" etc.

  - **Quality review process**. A quality review process would be needed but it is usually absent, and, in the rare cases it is present, it is not documented at all or poorly documented.
  - **Better peer reviews**. Some interviewees believe that OSS software should undergo peer reviews, but, when it does, peer reviews are usually not of adequate quality.

- **Quantitative evaluation methods to assess quality**.
  - **Benchmarks**. Benchmarks are usually absent, while benchmark description and results would often be very useful.
  - **Certifications**. Certifications of packages and systems should be done from an independent certification body. Linux products should conform to Linux standard base (LSB) standards, and should work on every Linux distribution.
- **Bug lists, bug management and bug statistics**. It would be useful to have tools that give feedback to the developers about bugs. Bug reporting tools should be integrated with the product. The history of each bug should be publicly available.
- **Usage**. It should be possible to answer questions like "Where and how the software has been used?", "How is the software perceived?", that is, some more information bearing than just "it works/it doesn't work." Usage collection tools should be integrated in the products. The number of downloads should be given, but is usually not enough.
- **Popularity and community sentiment**. To answer questions like "How many developers are working on a specific OSS project?", "Is this project considered interesting?" etc., interviewees suggest that some data should be collected from community lists, while others that data should be collected from CVS servers (both approaches are possible).
- **Updated and reliable documentation**. The most requested (and often missing) updated and reliable documentation is about architectural details, internals, and, more generally, process descriptions. In addition, process descriptions are useful to quickly join into the development of existing OSS software.
- **Specific documentation**. In general, the most requested documentation topics are: process documentation, development documentation, software architecture documentation, "getting started" documentation, configuration documentation and installation documentation. Interviewees also requested documentation translated into their language.
- **The relationship between the sponsor organization and the OSS**. Interviewees expressed the interest to be able to answer questions like "How much is the OSS product supported by the main sponsors?", "What is

the sponsors' policy with the OSS product?", "How are the sponsors organized in the development of the OSS product?", etc.

- **Presence of known companies**. The presence of known companies in the community of users, even if they are not sponsoring the project or if they are not directly involved in the development/maintenance.
- **Inline demo**. Useful to test the product before downloading and installing it.

5.5 Quality: Product Quality Issues When Choosing OSS

As already mentioned in Section 3.2.6, the ISO 9126 qualities [41] were chosen in the questionnaire as a reference for external quality factors. Usual internal product quality factors were also chosen.

The factors' name, mean importance rating and grouping are reported in Table 5-4. In the factors' names, iq stands for "internal quality," while eq indicates an "external quality."

Table 5-4: Quality factors: group and mean

section	*factor*	*group*	*mean*
Quality	functional requirements	8	8.609
Quality	eq reliability	7	8.082
Quality	eq performance	5	7.096
Quality	eq usability	5	7.000
Quality	eq maintainability	7	7.944
Quality	eq portability	4	6.310
Quality	iq size	1	3.926
Quality	iq complexity	3	5.696
Quality	iq modularity	6	7.456
Quality	iq standard architecture	6	7.368
Quality	iq patterns	3	5.882
Quality	standard compliance	6	7.357
Quality	self containedness	3	5.986
Quality	interoperability	7	7.931
Quality	localization	3	5.986

Not unexpectedly, functionality was almost unanimously taken as the most important quality. Factor *functional requirements* lies in group 8 (fundamental importance), which is the most relevant factors group, and contains this one factor only.

Some of the external qualities are believed to be very important: *eq maintainability* and *eq reliability* factors lie in group 7 (very high importance).

The remaining external qualities are also considered fairly important, since they all belong to group 5 and 4 (medium and fairly high importance): *eq performance* (group 5), *eq usability* (group 5) and *eq portability* (group 4).

In conclusion, the ISO 9126 [41] external qualities are considered quite important, and none of such factors falls below the 4th group (medium importance group).

As for code and design intrinsic qualities, the use of a standard architecture and a good modularization of the project are considered to be important, hence *iq modularity* and *iq standard architecture* both lie in group 6 (high importance).

However, the remaining internal qualities are not considered important, since *iq complexity* and *iq patterns* are in group 3 (low importance) and *iq size* is in group 1 (negligible importance).

Surprisingly, size is generally believed by the interviewees as unimportant. In the literature [51] and in a number of experimental studies, size is taken as the most important driver for a number of qualities of industrial interest, such as development effort, development time, and the number of faults.

We expected that product and design qualities would be regarded as relatively more important by developers, and process qualities by managers, but this view is actually not supported by our statistical analysis, as most of the associations between these factors are too weak to be statistically significant.

Factor *interoperability* is believed to be very important (it lies in group 7, very high importance, the second highest group): OSS products are supposed to heavily interact with a number of other pieces of software. Another factor associated with the issue of interaction among software pieces is the *standard compliance* (it lies in group 6, high importance).

Factor *self containedness* shows the same problems as *iq size* or *iq complexity*. The factor is believed to be fairly important in the literature [51], but the answers collected shows that it is not considered important. The factor lies only in group 3 (low importance), and this is unexpected because:

- OSS components usually require other components (building on top of existing components is the very philosophy of OSS), but this create complexities in the build process, and in the management of component dependencies. Hence *self contained* OSS products should be an advantage.
- The lack of *self containedness* may become a problem when using OSS products mandates the use of closed-source components. Many OSS licenses do not permit the distribution of OSS products with closed source products.

Localization and human interface (factor: *localization*) are believed to be of lower importance than other factors (the factor lies in group 3, low importance), even though this heavily depends on the application kind and, in general, on the specific problem domains.

5.5.1 Answers to open questions

Concerning the open question on what kind of information the interviewees would like to have on product quality that is not available, a number of answers were provided.

The interviewees stated several times that **product and design documentation** is a major issue: it should be available, but it is usually not.

Other issues were mentioned, as follows.

- **Ease**. Ease of use and ease of installation of the OSS product.
- **Documentation quality and accuracy**.
- **Certification of the OSS product**. It is important to have a rating by a benchmarking organization that evaluates all of the process and product aspects.
- **Test quality**. Every test should have an accompanying documentation; security tests should be always included.
    - **Regression test suites**.
- **External qualities**.
    - **Customizability**.
    - **Ease of installation**.
    - **Robustness**.
    - **Scalability**.
    - **Innovation level**.
    - **GUI**.
        - **Usability**.
        - **User friendliness**.
- **Internal qualities**.
    - **Data integrity**.
    - **Documentation**.
        - **Internal code**.
        - **Architecture**.
        - **Performance**.
    - **Code clarity, readability**.
    - **Modifiability**.
- **Documentation on stability**. OSS may be released even when it is not stable. This is the common maturity process of OSS, but is not suitable to business environments.

5.6 Customer: Customer Requirements

The factors related to customer requirements are usually believed to be very important, because they are usually mandated by the customers or by the law. The factor *customer satisfaction* lies in group 7 and hence it is considered very important. Another factor considered of very high importance is *interoperability issues* which lies in group 7 too.

Table 5-5: Customer factors: group and mean

section	*factor*	*group*	*mean*
Customer	customer satisfaction	7	8.043
Customer	interoperability issues	7	7.833
Customer	law	5	7.030
Customer	standard imposed	3	6.227

The factor *law* lies in group 5, hence it is considered of fairly high importance. The only factor related to customer requirements considered of a lesser importance is *standard imposed* that lies in group 3 (low importance).

5.6.1 Answers to open questions

Other answers were given when dealing with customer requirements.

- **Developer satisfaction**. The developer, and not only the end user, must enjoy the tool, so his work will be of much better quality.
- **Integration with related software**. It is important to let the customer understand how difficult it will be to integrate the chosen OSS product.
- **Opinions on the product by trusted and well known customers**. It is useful to build a collection of qualified opinions on the product by well known OSS community members.
- **Customer needs**. Customers sometimes have the complete control on the software requested. Thus it is the client that decides, among many other things, if closed source software or OSS software is to be preferred.

5.7 Associations between interviewees' characteristics and factors affecting their choices concerning OSS

The associations found and examined (see Table 5-6) show some interesting results. We will examine the associations for each characteristic of the interviewee (roles, education, etc.) with each factor (*TCO*, *ROI*, etc.).

- **Role: developer**. Developers show a slightly lower attention to *customer satisfaction* (the Developers mean is lower than the mean calculated on the non developers) but higher attention to respecting standards. Complying with standards is deemed important when considered from the quality point of view (*standard compliance*) as well as when considered from the customer point of view (*standard imposed*). Quite interestingly, developers show a higher attention on licenses (*type of licenses*) and a lower attention on the user community (*user community that witness quality*).

- **Role: project manager**. Project managers, in opposition to developers, show a higher attention to *customer satisfaction*, but they also highly appreciate the support of the product, as shown by factors *user community* and *short term support*.
- **Role: upper management**. Upper managers consider *interoperability issues* and *standard imposed* as slightly less important than average.
- **Type of organization**. No-Profit organizations show a lesser attention to economic factors (*ROI*), Public organizations show a lesser attention to *functional requirements,* and Private organizations, as it could be expected, show a higher interest on economic factors (the economic factor considered is the *ROI*).

For more information about the associations found, and the statistical tests used to find them, see section 9.1.2).

Table 5-6: Associations. "mean (factor = level)" is the mean calculated on the set of interviewees that have factor = level, "mean (factor ≠ level)" is the mean calculated on the set of interviewees that have factor ≠ level

factor	*level*	*mean factor*	*mean (factor = level)*	*mean (factor ≠ level)*
role developer	Yes	customer satisfaction	7.286	8.548
role developer	Yes	standard compliance	8.033	6.850
role developer	Yes	standard imposed	6.926	5.744
role developer	Yes	user community that witness quality	6.400	7.500
role developer	Yes	type of licenses	7.833	6.049
role project manager	Yes	customer satisfaction	9.308	7.754
role project manager	Yes	user community	8.714	6.950
role project manager	Yes	short term support	8.143	6.721
role upper management	Yes	interoperability issues	7.174	8.143
role upper management	Yes	standard imposed	5.045	6.818
type of organization	No_Profit	ROI	3.167	6.629
type of organization	Private	ROI	6.891	3.923
type of organization	Public	functional requirements	7.333	8.737

6 Literature Review

As already mentioned, the objective of the work reported here was:

- to collect information concerning how different types of organizations (Private, Public Administrations, No Profit, Education Institutions, etc.) perceive trustworthiness;
- to clarify these organizations' goals when dealing (developing, using, integrating, customizing, etc.) with OSS trustworthiness issues;
- to understand what are the trustworthiness-related factors that influence their decisions about OSS.

The work was based mainly on the collection of data through questionnaires and interviews, which resulted in a fairly large amount of new information. Nevertheless, the existing literature also provides interesting information, so, the analysis of the data collected through questionnaires is supplemented by insights we gained based on the existing literature. In this section, we summarize the results of the survey we performed on the literature concerning OSS trustworthiness.

For each relevant topic (trust concept, trustworthiness concept, trustworthiness importance, trustworthiness evaluation models) we give a summary of the published information, and, finally, compare the literature points of view with the point of view expressed in the present document.

6.1 About the concept of software trust

Trust is a complex phenomenon that has been the object of interest in various disciplines. Depending on the approach, trust has been defined in many ways.

As a consequence, we cannot take for granted the meaning given to the word "trust" when applied to OSS software and products.

It is therefore interesting to look at the concept of Trust in OSS as it emerges from literature.

6.1.1 Trust in communities

Antikainen [22] argues about the correlation between communities' sentiments and trust. She starts by assuming that trust is a key factor in communities' discussions, because someone may have an opportunistic behavior and so it may manipulate the public opinion about an OSS product positively or negatively. In other words, the public opinion can be influenced by incomplete, biased, or even incorrect information; this results in a changed public perception of the trustworthiness of an OSS product. Also, trust is a very important factor when organizations and companies are making decision about whether they choose an OSS product or not. Antikainen defines trust as "the extent to which a person is confident in and willing to act on the basis of, the words, actions, and decisions of another". Trust requires a relationship between a trustor and a trust target. She analyzes one of the most active communities on the OSS world: the Linux Kernel community. She found eight factors which seem to affect trust in the community, ordered by their importance: skills (the most important one), practices, reputation, common goals, sharing information, culture and values, possibility to influence, familiarity.

In the same line as Antikainen's, Hertzum aims to explain the trust value of the relationships between colleagues [3]. The trustworthiness of received information is an issue and Hertzum notices how important and easy it is for employees to ask colleagues for information rather than external sources. In relation to human to human interaction, trust is defined as an emotive issue where the trusted party has a moral responsibility toward the trusting party. To the trusting party, trust involves an assessment of whether the other person possesses the required knowledge and skills and is likely to give a truthful and unbiased account of what he or she knows. People place trust in each other to varying degrees, depending on several situational factors. It is possible to distinguish four types of trust by means of the evidence on which trust is founded and with respect to the amount of evidence involved:

- first-hand experience;
- reputation;
- simple inspection of surface attributes;
- general assumptions and stereotypes.

Thus, knowing an information source first-hand, or knowing someone who knows it first-hand, provides people with a more solid basis for assessing the trustworthiness of the source.

The results of the work by Antikainen [22] and Hertzum [3] are very well in line with the goals of the QualiPSo project, as the importance of trust is clearly highlighted. At the same time, it is seen as dangerous to rely only on the confidence between the 'client' and the trustee. A more systematic and reliable notion of trustworthiness is needed, as well as a technique to evaluate trustworthiness.

The factors considered in the questionnaire address subjective and objective factors. They investigate subjective aspects of trustworthiness as the reputation of the OSS as well as much more objective quality factors (such as size and complexity of a software product). Hence, the questionnaire and the factors collected are quite in agreement with the concept of trust for communities found in literature, since according to the literature both objective and subjective factors should be taken in consideration.

6.1.2 Defining trustworthiness

Hansen et al. [24] observe that security and privacy can be generally stated in an objective way, while trustworthiness strongly depends on the subjective experience and feelings of the user. Therefore, they define trustworthiness on the reliable basis of the concepts of security and privacy.

While qualities such as integrity or availability can be formulated as "do's" and can be verified by practical tests, privacy requirements are very often "don'ts". For instance, the main security goal of privacy is confidentiality, which is clearly a "don't", the "don't" clearly refers to the publishing of sensitive information.

Such requirements, as well as the proof of "don'ts" in general, can only be validated by disclosing the source code. To this extent, OSS is at a great advantage with respect to closed source software.

Hasselbring and Reussner [25] aim to provide a holistic view of trustworthiness in software in an interdisciplinary setting. In their view, trustworthiness consists of the following attributes: correctness (the absence of the improper system states), safety (the absence of catastrophic consequence in the environment hosting the system), quality of service (availability, reliability, performance), security (the prevention of unauthorized access to the system), privacy (the absence of unauthorized disclosure of information).

In [26] Lawrieand and Gacek present issues raised by the articles, presentations, and discussions concerning Open Source Software, Trustworthiness, and Dependability at the Open Source Development Workshop held in Newcastle upon Tyne, UK, in February 2002. Among other contributions, they underline some key concepts about OSS and trustworthiness. Firstly, they assert that the terms Trustworthiness and Dependability are equivalent. They also report some considerations and definitions of trust and trustworthiness. Trust may exist when there is no evidence to justify the reliance on a given system, whereas trustworthiness suggests that there are assurance criteria to justify our confidence in a system. To be a dependable and trustworthy system, a computer system needs to include certain attributes such as security, reliability, availability.

Bernstein [5] analyzes how rarely trustworthiness (of both OSS and closed source software) is taken into consideration by software designers, especially with respect to issues such as schedule, cost, performance, and requirements. Bernstein complains about the lack of interest around trustworthiness and advocates laws that require that every software product report the names of a Software Architect and a Software Project Manager who guarantee the trustworthiness of the product and of the development process. Trustworthiness is a holistic property, encompassing security, safety and reliability. It is not sufficient to address only one or two of these diverse dimensions, nor is it sufficient to simply assemble components that are themselves trustworthy. Integrating the components and understanding how the trustworthiness dimensions interact is a challenge. Because of the increasing complexity and scope of software, its trustworthiness will become a dominant issue.

Bernstein [5] also states that software fault tolerance is at the heart of building trustworthy software. Trustworthy software is stable software. It is sufficiently fault-tolerant that it does not crash at minor flaws and will shut down in an orderly way in the face of major trauma. Trustworthy software does what it is supposed to do and can repeat that action time after time, always producing the same kind of output from the same kind of input. The United States' National Institute of Standards and Technology (NIST) defines trustworthiness as "software that can and must be trusted to work dependably in some critical function, and failure to do so may have catastrophic results, such as serious injury, lost of life or property, business failure or breach of security".

From the analysis of the literature, it is clear that for OSS the notion of trustworthiness encompasses all the factors that contribute to the decision whether OSS can be sufficiently trusted to be used, or included in a product, especially as an alternative to commercial closed-source software. That means that the factors to be taken in account to evaluate the trustworthiness of an OSS product should include factors as indicated by Bernstein [5] (such as dependability, safety, etc.), which are used as well to assess closed source software, as well as OSS specific factors (such as factors to assess the community behind an OSS product, its level of maturity, etc.).

6.2 The Importance of Being Trustworthy

In this section we examine some peculiar situations of OSS adoption and perceived OSS trustworthiness. Australia's situation is particularly interesting, being one of the few countries to have almost completely rejected the adoption of OSS products. A Canadian study on OSS is examined, primarily to find out commonalities with our approach, and also to have a clear picture of the OSS situation in Canada and to understand the approach used by the Canadian government to clarify the OSS situation. Finally, the Italian Public Administration situation is examined, where some hindrances and obstacles with OSS adoption in Public Administration are thought to be of a political kind.

6.2.1 The situation in Australia

Goode's survey [27] reports an in-depth analysis of a surprising en mass rejection of OSS by Australia's top firms. The survey was made on a sample of 500 companies. The study found that managers rejected open source software because they could not see that it had any relevance to their operations, perceived a lack of reliable ongoing technical support of it, and also seemed to foresee substantial learning costs or had adopted other software that they believed to be incompatible with open source software.

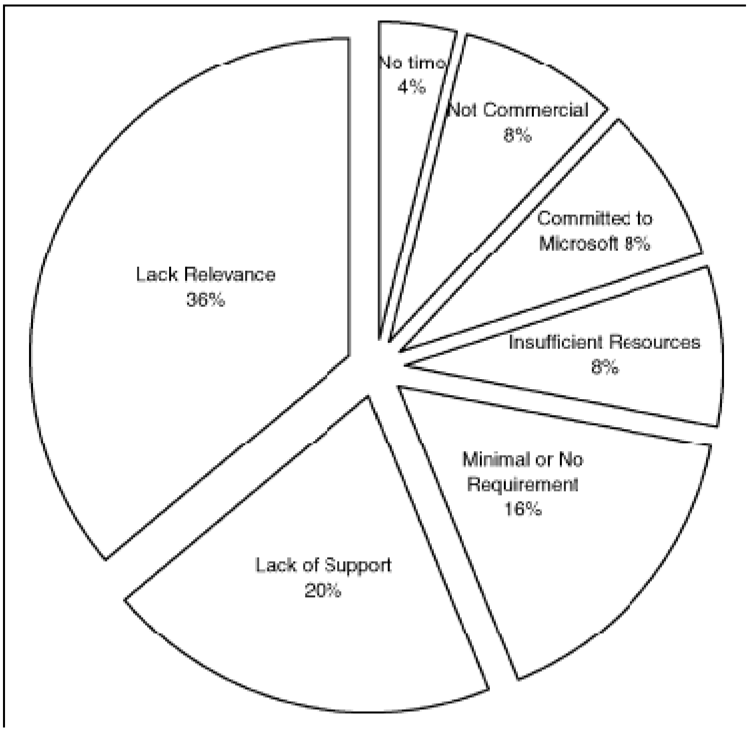


Figure 6-1: Reasons for rejecting OSS

Figure 6-1 reports the percentages breakdown of reasons for OSS rejection. The main reasons are the following:

- Lack of Relevance. Most respondents had perceived only little relevance of OSS to their business, and could not see any benefits to use it. Some respondents argue that they might be open to adopt it in the future. "One firm argued that they had not adopted OSS because other nearby firms had rejected open source software. This suggests that, for at least some managers, peer information networks are significant." This also confirms the high relevance that trustworthiness has in peer communications, as already indicated by Hertzum [3].
- Lack of Support. The second largest segment cited a lack of conventional and ongoing support as a critical factor in their decision not to adopt OSS products. Here are some quotes from the interviewees. "We think there's a real lack of tangible support."; "We're not interested because it's not a commercial offering."; "We really don't know anything about them and don't want to know. We want someone we can sue when things go to the wall".
- Requirement. The next group had evaluated open source technology but had determined no business requirement for it: "at the moment it's just not feasible - we have no requirement for it". This suggests that managers might be poorly exploring existing software models. Although a huge variety of OSS is proposed to companies, managers would rather stay with their closed source offerings.
- Resources. A number of respondents noted a lack of time and resource (i.e., companies and managers do not have enough time and/or resources to invest in OSS) as the barriers to open source software. Summarizing in one

sentence, quoted by [12]: "open source software is only free if your time has no value".

- Committed to Microsoft. This is an interesting percentage (8%). The interviewees assert that the committing to Microsoft precludes them for making use of OSS.

6.2.2 The Canadian Collaborative Fact Finding Study

The main aim of the Canadian Collaborative Fact Finding Study [28] was to raise the level of understanding of why and how the OSS paradigm and its products, services and communities are important to Canada, both domestically and internationally. The report tries to fill a lack of information on OSS awareness, initiatives, opinions and attitudes in Canada. The study includes (quoting from the text):

- A scan and review of commercial and non-commercial open source business models for software, applications and services delivery, to identify recent trends in Canada, the United States and other major markets, and the most credible forecasts of future trends.
- Industry profiles of key ICT suppliers in Canada who support or supply open source software, applications and/or services.
- An assessment of the engagement of business, government, academia and civil society organizations in Canada toward OSS products, in order to better understand awareness, concerns about support and liability and conditions for acceptance.
- Assessment of the business advantages of alternative open source software licenses and marketing strategies, from the standpoint of both suppliers and users.
- A synthesis of the issues, opportunities and constraints for Canadian industry and government decision-makers.

The e-Cology Corporation organized the methodology which this study was delivered with. First, they exhaustively surveyed all the Canadian and international literature published on OSS. Subsequently, a workshop on the future of software and OSS in Canada was held in Ottawa. After the workshop, Canadians were invited to answer an online questionnaire. The Corporation obtained more than 180 responses to be analyzed. Finally, 17 Canadian companies active in OSS business had been profiled to produce fact sheets on their products and services. The diagram in Figure 6-2 presents a composite view (depicted from a technology diffusion model developed by Industry Canada and here adapted and applied to facilitate an high level interpretation of the study results) of the state of OSS in Canada based on the primary research findings.

Open source adoption is framed in the context of its Political, Market and Infrastructure Environmental factors, which determine the starting conditions, and ongoing forces, which influence adoption of open source. Among other results, the study reveals how *trust and collaboration are the DNA of OSS*. In fact, OSS requires a very deep understanding of the dynamics, conditions and beliefs in the power of collaboration.

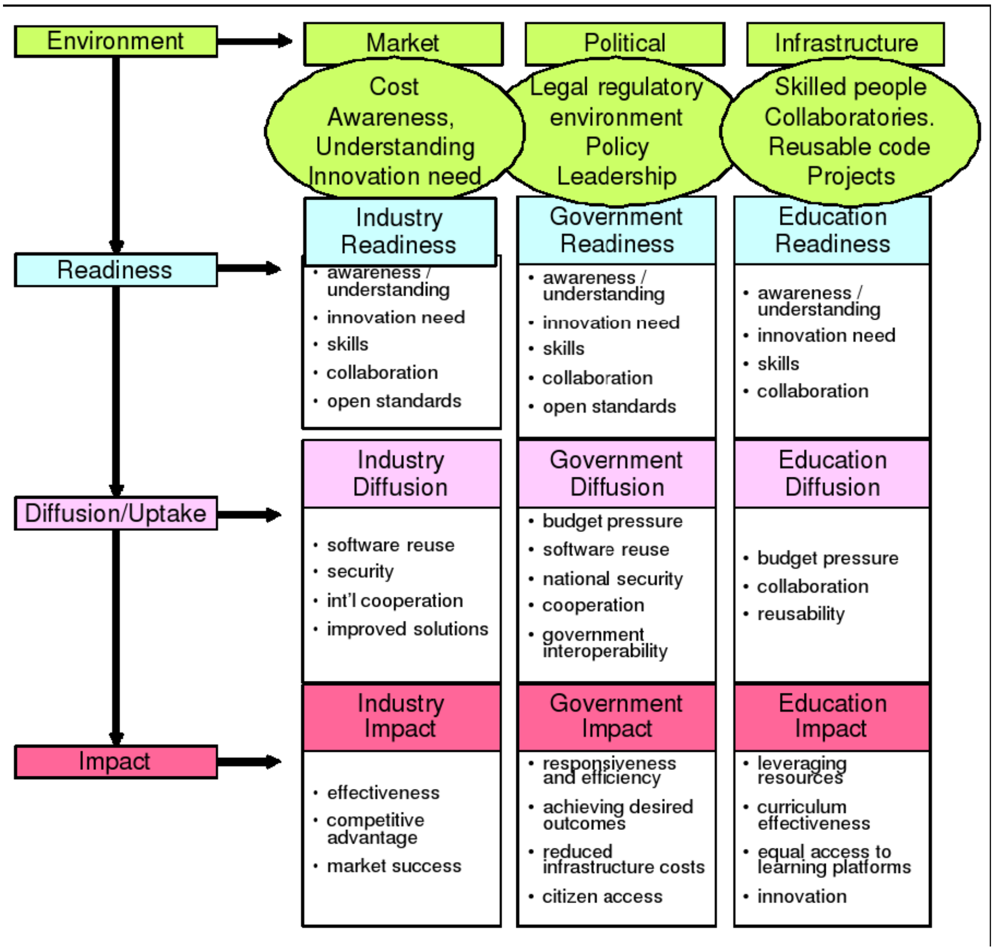


Figure 6-2: An Overview of OSS with respect to Readiness, Uptake, Impact.

6.2.3 The Italian Public Administrations and OSS

The sentiment of Italian PAs on OSS is contrasting. On the one hand, many offices use multiple hardware/software platforms (Windows XP but also MacOS, Ubuntu, SuSe, RedHat or AIX), as desktop, servers, data management, front-end systems. But there is still distrust from PAs towards OSS alternatives [30]. On the other hand, in June 2007, the Ministero per l'Innovazione e le Risorse nella PA has founded the "OpenSource" Commission, composed of several of the main Italian experts. At the same time the Open Source Observatory was started [43], hosted by CNIPA [42] (National Centre for Informatics in the Public Administration); one of the first objective was to shed light over reuse aspects of software products [44][45]. There is also an initiative fulfilled by the Roma Linux User Group. The project OpenPA [46] aims to spread the OSS knowledge toward PAs and schools. The Regione Piemonte has built the Consorzio per il Sistema Informativo [47] to promote innovation in PAs using the most recent ITC technologies. This Consortium has eight local offices and 54 members. The Consortium trusts in OSS and it has used OSS for 10 years. During 2006 it has launched an OSS middleware platform, named OASI (Open Available Secure Integrated) [48], to develop and provide services to PAs and users. Ancitel S.p.A. has renewed its platform investing in OSS projects. Ancitel provides technological services to Italian municipalities, having as technological partners ACI IT division and Telecom Italia S.p.A.. ACI itself is supporting six different projects for PAs.

Regione Piemonte is still one of the more active subjects in the adoption of OSS software. There are two remarkable projects. Strategie Digitali S.r.l. has chosen to use only OSS for its services and products [49]. They aim to reach a more extended ROI, to have a social feedback, to reduce the "digital divide". Companies and PAs can use spared money thanks to the non-existent cost for OSS licenses investing them towards education, personalization, information updating, and evolution. The other project, named OSS Piemonte [50] and funded by Regione Piemonte, gathers a set of companies which collaborate to achieve the objective of using OSS solutions to provide services and products to their customers.

6.2.4 The Importance of Being Trustworthy: the QualiPSo view

The Australian analysis reports boils down to: there is a lack of understanding of OSS trustworthiness, and there is a lack of tools and methods to assess the trustworthiness of OSS. The Australian case highlights the need to have more reliable and structured information on OSS products, such as (but not limited to) formalized trustworthiness evaluation processes, certifications, OSS structured and detailed catalogues (where each OSS product is described, and a specific evaluation is provided taking in consideration the eventual specificities of the environment, such as specific laws, standards to be adopted, etc.).

Consistent with the findings of the Australian analysis, the Canadian initiative demonstrates that the understanding and promotion of OSS needs an effort devoted to understand a set of issues concerning OSS: business models, regulations, skills and resources usage and development, etc.

Finally, the Italian situation demonstrates that the diffusion of OSS in the Public Administration is increasing, provided that it is effectively supported by initiatives that help the administrations in understanding the application conditions and tradeoffs of OSS.

The reports that have been analyzed emphasize the need to better understand OSS trustworthiness and to develop methodologies and tools to assess OSS trustworthiness. Besides, they also indicate (especially the Canadian report [28]) factors that should be considered when assessing OSS trustworthiness, and in fact have been considered in QualiPSo questionnaire. There is not a perfect one to one mapping, but most of the factors worth considering in the reports can be traced back to one or more questionnaire factors.

6.3 OSS evaluation models and tools

There is a general uneasiness with OSS, in order to overcome some of the difficulties encountered when adopting OSS, several OSS evaluation models and tools have been developed. Their aim is to help potential adopters to understand the characteristics of the available products, and to evaluate the pros and cons of its adoption.

In this section, some of the best-known OSS evaluation models are analyzed.

6.3.1 OSMM, Navica

The OSMM (Open Source Maturity Model) is designed to enable organizations to evaluate OSS products and understand whether a product can fulfill the organization's requirements [31].

Companies, as well as PAs and organizations, often wonder whether an open source product will satisfy their needs. The OSMM method evaluates an OSS product by assessing its support, training, documentation, integration and offered services. These are the main requirements a company has to have satisfied in order to adopt a software product. OSMM comes with a recommended minimum maturity scores to give a context to compare to the new evaluations.

OSMM assesses product maturity in three phases:

1. Assess vital product elements (software product, support, documentation, training, product integration, professional services) for maturity and assign a maturity score between 0 and 10.
2. Define a weighting for each element based on the organization requirements. The overall maturity score at the end of the final step will be normalized to a 100 point scale.
3. Calculate the product overall maturity score. The element scores are summed to give an overall product maturity score on a scale of 1 to 100.

6.3.2 OSMM, Capgemini

Capgemini developed an Open Source Maturity Model in seven steps to allow organizations, PAs and companies to determine if or which OSS product is suitable [32]. The Capgemini OSMM describes how an Open Source product should be assessed to ensure that the product meets the IT challenges companies face today. Twenty seven OSS indicators have been found, either for products and applications.

Product indicators are important in having a number of objective and measurable facts. To assess also the context in which a product grows and to place the product in its context giving it a score, it is necessary to have the application indicators. Product indicators are twelve and are grouped in four groups:

- Product
  - Age, Selling points, Developer community, Human hierarchies, Licensing
- Integration
  - Collaboration with other products, Modularity, Standards
- Use

  - Support, Ease of deployment
- Acceptance
  - User community, Market penetration

The application indicators are fifteen and they are: usability, interfacing, performance, reliability, security, proven technology, vendor independence, platform independence, support, reporting, administration, advice, training, staffing, and implementation. For each indicator, the client gives a score on a 1 to 5 scale, 1 being ‘not important’ and 5 being ‘extremely important’.

The seven steps required by Capgemini to assess a product using product and application indicators are the following ones.

1. Product research and rough selection.
2. Scoring of products using the product indicators.
3. Scoring using application indicators (by a Capgemini consultant).
4. Interview with customer on the value (importance) of the application indicators.
5. Scoring the application indicators by the customer (together with Capgemini consultant).
6. Determining score card per product and final selection of right product for customer (and for Capgemini).
7. Evaluation.

This model evaluates more than one product at a time and the set of products initially selected against the application indicators.

6.3.3 OpenBRR

OpenBRR.org proposes a model, named Business Readiness Rating for Open Source, as an open standard to facilitate assessment and adoption of OSS [29].

They point out how, in practice, many software evaluation projects are done ad-hoc, without a formal assessment methodology. Ad-hoc methods may be incorrect or incomplete in their assessment, and it is extremely difficult to validate the correctness of the evaluation. They suggest that using an open (to promote trust in the assessment process) and standard (to allow common understanding of the assessment ratings) model to assess software will increase the ease and correctness of evaluation, and accelerate the adoption of open source software. Additionally, OSS users can share their assessment result with OSS communities.

On the official Open BRR site several evaluations are available. They can be examined and easily adapted: one just needs to input the parameters that suit best one’s own needs in the spreadsheet containing the evaluation. The proposers of the method plan to apply it to all SourceForge and Java.net projects, so that potential users can find a ready to use evaluation of the software they are interested into.

In the first step of the evaluation, the list of programs to be evaluated is compiled. Then every component is evaluated with respect to a set of indicator selected according to the target usage and including: the type of license, the compliance with standards, the existence of a user base, the availability of reliable support, the implementation language, internationalization, etc. Then the functionality of products is evaluated. The features of a "reference application" are identified and their importance is graded with respect to "standard usage". Then every product is evaluated with respect to how well it implements every feature. Finally, the grades are normalized and the final evaluation (a grade in the 1...5 range) is computed.

The Open BRR is a relevant step forward with respect to the OSMM, since it includes more indicators, the idea of the target usage, and the possibility to customize evaluations performed by other, just by providing customized weights. With respect to the latter characteristics, the Open BRR has however some limits: 1) for many products, it is difficult to choose a "reference application" that reflects the needs of all the users; 2) there are many possible target usages, each with its own requirements; 3) finally, every subjective evaluation performed by a user could be not applicable to other users. In any case, the final score is probably a too synthetic indicator to represent the complex set of qualities of a software product.

6.3.4 QSOS

QSOS (Qualification and Selection of Open-Source software) is a free method developed by Atos Origin to allow software qualification by integrating the open source characteristics and software comparisons according to formalized needs requirements of weighted criteria, in order to make a final choice [30].

The general process of QSOS is made up of four interdependent steps (see [30]):

1. The definition phase aims at identifying the factors to be considered in the following phases.
2. The evaluation phase aims at collecting the relevant information concerning the products from the OSS community. The goal is to create an identity card (IC) for every product with general information, available services, functional and technical specifications, etc. The quality aspects of the selected products are evaluated and a grade (in the 0...2 range) is assigned according to the evaluation guidelines provided by QSOS.
3. The qualification phase is dedicated to the definition of the selection criteria. The user's needs and constraints are described.
4. The selection phase consists in the comparison of the products' evaluation forms with the selection criteria, and in the identification of the product that matches betters with the user's needs and constraints.

The main contributions of QSOS probably are: 1) making explicit the set of characteristics that compose the IC, and 2) the provision of a guideline for the consistent evaluation of these characteristics. Nevertheless, the evaluation procedure is too rigid and a bit cumbersome. Finally, even though in the selection criteria it is possible to classify requirements as needed or optional, there is no proper weighting of features with respect to the intended usage of the software.

6.3.5 OpenBQR

OpenBQR (Open Business Quality Rating) [33][34] merges and extends OpenBRR and QSOS. It introduces new evaluation criteria and overturns the steps of selecting and of weighting products, starting from the weighting of elements and then, basing on the weight, evaluating which elements have to be scored. OpenBQR aims to be an open, standard, adaptable, complete, simple model. OpenBQR assessment process can be done in three steps:

- Quick assessment filter.
- Data collection and processing.
- Data visualization.

Like in OpenBRR OpenBQR in the first step identifies a list of element to evaluate. Unlike other models, OpenBQR first assigns a weight for every element considering five indicators areas:

- Product use target (mission-critical, regular, development, experimentation, but also the license type, the standards compliance, the implementation language, the internationalization support are all parameters that can be considered).
- Internal qualities analysis (starting from ISO 9126 - "Information Technology - Software product evaluation - Quality characteristics and guidelines for their use").
- External quality analysis (e.g., using a bugs database).
- Support availability in time (considering, e.g., the team’s size, releases umber published every year, the presence of external plugins).
- Evaluation of functional requisites.

The second step starts by deleting all the elements where weight is zero or near zero. For every area, the weights are normalized and a score based on the importance of the element is set. Finally, every weight is multiplied for the value of the score, obtaining a final result for every area. The final score for every product can be achieved summing all the product scores for every area.

The final step provided by OpenBQR method is data visualization, with a grid reporting the results for every product.

6.3.6 OSS evaluation models and tools: the QualiPSo view

The OSS evaluation models illustrated above indicate that there is an increasing availability of support for OSS potential adopters. The organizations that developed these evaluation models appear to have -at least partially-understood the needs of trustworthiness-related information on which the adoption of OSS have to rely.

However, the illustrated models have two limitations:

- They address mostly technical issues, often leaving out important aspects concerning economic and legal issue as well as other aspects concerning the processes of acquisition, development, integration, etc.
- They remain at the stage of proposals. In practice, people who are interested in OSS do not use these methods (most people do not even know about them).

The limitations suggest some remedies:

- A broader concept of trustworthiness should be assumed. The results of the investigations reported in this document confirm that only some specific technical qualities contribute significantly to the trustworthiness as perceived by OSS developers and users. Instead, several non-technical qualities are counted among the factors that contribute to determine the level of trustworthiness of a product. Among these, our analysis indicate economic issues (through the ROI factor), legal issues (through the *law* and *type of licenses* factors), and organizational/technical issues (through the *standard compliance* and *standard imposed* factors) as fairly important (see Table 5-1).
- The awareness of trustworthiness-related issues in the software developers and users communities should be increased by means of proactive dissemination activities.

The aspects used in the evaluation methods which relate to trustworthiness are fairly well represented in our questionnaire. For instance, OSMM Navica [31] defines 6 broad evaluation elements which can be related to certain factors e.g. listed in Table 5-1:

- Software: all factors from section Quality
- Technical support: community related factors
- Documentation: factor *documentation*
- Training: factors *training / guidelines*, *TCO*
- Integration: factors *interoperability*, *interoperability issues*
- Professional services: not well represented, but somehow connected to community related factors

OSMM Capgemini defines much more elements to evaluate (27). Most of these elements are represented in the Questionnaire, with a few exceptions (human hierarchies, market penetration, vendor independence, platform independence, reporting).

7 Conclusions

The objectives of the research presented here are quite straightforward: 1) to find out which are the factors most commonly used to assess the trustworthiness of an OSS product, 2) to understand how these factors correlate to each other and to the types of organizations, to the types of OSS uses, and to the profile of the specific interviewee.

The achievement of these objectives is very important for the following activities of QualiPSo activity A5 (Trustworthy Results). Specifically, the findings and insights of the present research will guide and be the basis of the following tasks: GQM plans, identification of measures, requirements definition for software tools and tool development, and validation of the identified factors on the field.

For this purpose, we want our research to have very strong pragmatic foundations, to address real needs and solve real problems. Accordingly, we want to provide an OSS trustworthiness definition based on the actual perspectives that OSS users have.

The results extracted from the questionnaires by means of the statistical analysis show interesting results concerning OSS trustworthiness. Some of these results confirm and give evidence to support previous beliefs, while others are surprising and unexpected. The set of interviews is not huge (103 interviews); yet it allows the achievement of statistically significant results.

Several indications obtained through the interviews confirmed the expected indications. Quite noticeably, most of the expected indications involve technical issues. The factors that directly or indirectly involve the user requirements (whoever the user may be) have been found to be considered very important, as well as interoperability and standard compliance factors. The community is the most important factor used to understand an OSS project's vitality, health, actual usability, and potential longevity. External qualities are also believed to be important when assessing the trustworthiness of an OSS project. Finally, documentation of almost any kind is considered very important when choosing an OSS project.

Several unexpected evaluations emerged:

- *Complexity and size*. Two of the most widely used and accepted attributes of software systems are complexity and size. Nevertheless, these are considered of very low importance in the interviews (especially size, that is the least important factor of all).

  The reasons for this common opinion are not completely clear. A possible explanation (which needs further validation) is that a complex software system with a large community base is preferred to a small one with no community, because an OSS project will be understood with the active help of the community.

- *Economic factors*. ROI and TCO are considered only fairly important, far from being very important as it was expected (and as is widely publicized).

Traditionally, the main leverage for promoting the adoption of OSS has been the economy of the approach, i.e., the software being available at no immediate cost. The advantages of OSS from the economic point of view were widely proved as far as ROI is concerned, while they were much debated as far as TCO is concerned. This situation seems to be changing, and the economic factors are no longer perceived as the leading factors.

- *Licenses*. Licenses and law factors are considered to be quite important, but not as important as it could be guessed. There is a clear agreement on GPL-like licenses. Some (but not all) of the OSS users understand very well the long term advantages of GPL-like licenses against totally permissive licenses like the BSD free license. In other words, the need for licenses that allow a user to take, use, and redistribute OSS code as he or she pleases is less important than expected.

The work reported here yields new findings on OSS and how its trustworthiness is perceived in industry. However, the main contribution of the information reported here is to provide a solid base for the following tasks of activity A5 - Trustworthy Results, i.e. to build a model of OSS trustworthiness, to be used whenever this kind of indication is needed: e.g., when users select the software to be used and want to ascertain that the chosen SW is trustworthy, or when developers plan and carry out the implementation of OSS and want to ascertain that their product can be trusted by the potential users.

8 REFERENCES

[1] Amoroso, E., Taylor, C., Watson, J., and Weiss, J. "A process-oriented methodology for assessing and improving software trustworthiness". In Proceedings of the 2nd ACM Conference on Computer and Communications Security, pp. 39-50, 1994.

[2] Neumann, P. G., "Robust Nonproprietary Software", In Proceedings of the IEEE Symposium on Security and Privacy, pp. 122-123, 2000.

[3] Hertzum, M. "The importance of trust in software engineers’ assessment and choice of information sources". In Information and Organization, vol. 12, no. 1, pp. 1-18, 2002.

[4] "Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU". UNU-Merit report, 2006.

[5] Bernstein, L., "Trustworthy software systems", ACM SIGSOFT Software Engineering Notes, vol. 30, no. 1, pp. 4-5, 2005.

[6] Dustin, E., "The Software Trustworthiness Framework", Zero in a Bit, 30 January 2007.

[7] Fuggetta, A., "Open source software, an evaluation", The Journal of Systems and Software, vol. 66, pp. 77-90, 2003.

[8] Wheeler, D. A., "Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers", dwheeler.com Report, 2007.

[9] Fisher, R. A., "On the interpretation of $\chi 2$ from contingency tables, and the calculation of P", Journal of the Royal Statistical Society, vol. 85, no. 1, pp. 87-94, 1922.

[10] Abdi, H., "Binomial Distribution: Binomial and Sign Tests", in Encyclopedia of Measurement and Statistics, Thousand Oaks, 2007.

[11] Mann, H. B., Whitney, D. R., "On a test whether one of two random variables is stochastically larger than the other", in Annals of Mathematical Statistics, vol. 18, pp. 50-60, 1947.

[12] Mood, A. M., "Introduction to the Theory of Statistics", third edition, McGraw-Hill, 1974.

[13] The Open Source Initiative. Internet Website. Accessed on June 2007. Available online via http://www.opensource.org/.

[14] The Free Software Foundation. Internet Website. Accessed in June 2007. Available online via http://www.fsf.org.

[15] NIST (National Institute of Standards and Technology) HISSA (High Integrity Software Systems Assurance). Internet Website. Accessed in June 2007. Available online via http://hissa.nist.gov.

[16] Raymond, E.S., "Goodbye, free software; hello, open source", internet website, accessed in June 2007, available online via http://www.catb.org/esr/open-source.html.

[17] Raymond, E., "The Cathedral and the Bazaar", FirstMonday, Vol.3 N.3 - March 2nd. 1998, available on line via http://www.tuxedo.org/~esr/writings/cathedral-bazaar/.

[18] European Working Group on Libre Software, Internet Website, accessed in June 2007, available online via http://eu.conecta.it.

[19] European Commission. "Free/Libre and Open Source Software: Survey and Study", Internet Website, accessed in June 2007, available online via http://www.infonomics.nl/FLOSS.

[20] Cerri, D. and Fuggetta, A., "Open Standards, Open Formats, and Open Source." CEFRIEL, Politecnico di Milano, version 5.0 final draft, January 2007.

[21] Fuggetta, A., "Open Source and Free Software: A New Model for The Software Development Process?", UPGRADE, European Journal for the Informatics Professional, Vol. V, No. 5, October 2004.

[22] Antikainen M., "Is trust based on cognitive factors in OSS communities?", Trust in Open-Source Software (TOSS) 2007, Limerick, June 2007.

[23] German D. M., Gonzales-Barahona J. M., Robles G., "In what do you trust when you trust? The importance of dependencies in trust analysis", Trust in Open-Source Software (TOSS) 2007, Limerick, June 2007.

[24] Hansen M. Ks., Úhntopp K., Pfitzmann A., "The Open Source Approach - Opportunities and Limitations with Respect to Security and Privacy". Computers & Security, vol. 21/5, pp. 461-471, 2002.

[25] Hasselbring W., Reussner R., "Toward Trustworthy Software Systems", US Army Research Laboratories Information Assurance Center, IEEE. 2006.

[26] Lawrieand T., Gacek C., "Issues of Dependability in Open Source Software Development", ACM SIGSOFT, Software Engineering Notes, vol. 27, n. 3, pp 34, May 2002.

[27] Goode S., "Something for Nothing: Management Rejection of Open Source Software in Australia’s Top Firms", Information & Management, vol. 42, pp. 669 - 681, 2005.

[28] "A Collaborative Fact Finding Study. Open Source Business Opportunities for Canada’s Information and Communications Technology Sector", e-Cology Corporation, available on-line via http://www.e-cology.ca/canfloss/report, 2003.

[29] "Business Readiness Rating for Open Source", OpenBRR.org, BRR2005 - RFC 1, accessed in June 2007, available on-line via http://www.openbrr.org.

[30] "Method for Qualification and Selection of open Source software (QSOS)", Atos Origin, version 1.6, accessed June 2007, available on-line via http://www.qsos.org.

[31] Golden B., "Making Open Source Ready for the Enterprise: The open Source Maturity Model", extracted from "Succeeding with Open Source", Addison-Wesley, 2005.

[32] Duijnhouwer F. W., Widdows C., "Open Source Maturity Model", accessed June 2007, available on-line via http://www.seriouslyopen.org.

[33] Taibi D., Lavazza L., Morasca S., "OpenBQR: a framework for the assessment of OSS", The Third International Conference on Open Source Systems, Limerick, Ireland, 11-14 June 2007.

[34] Taibi, D., "Open Business Quality Rating (OpenBQR)", degree thesis at Università degli Studi dell’Insubria, 2006, accessed on June 2007, available on-line via http://www.taibi.it/OpenBQR/.

[35] Ruffatti G., Oltolina S., Tura D., Damiani E., Bellettini C., Colombo A., Frati F., "New Trends Towards Process Modelling: Spago4Q", Trust in Open-Source Software (TOSS), Limerick, June 2007.

[36] Weiss D., "Measuring Success of Open Source Projects Using Web Search Engines", in proceedings of the Open-Source Software Conference (OSS) 2005.

[37] Hahn R.W. (editor), "Government Policy toward Open Source Software", 2002.

[38] Krishnamurthy S., "Cave or Community? An Empirical Examination of 100 Mature Open Source Projects", 2002.

[39] Stamelos I., Angelis L., Oikonomou A., Bleris G., "Code quality analysis in open source software development", Systems J, volume 12, pp. 43-60, 2002.

[40] "Object Management Group (OMG)", Internet Website, accessed in June 2007, available on-line via http://omg.org/.

[41] "ISO/IEC-9126:2001 Information technology, Product Quality", International Standard ISO/IEC 9126, International Standard Organization, June, 2001.

[42] "Centro Nazionale per l'Informatica nella Pubblica Amministrazione (CNIPA)", Internet Website, accessed in June 2007, available on-line via http://www.cnipa.gov.it.

[43] "Osservatorio Open Source (OOS)", Internet Website, accessed in June 2007, available on-line via http://www.osspa.cnipa.it.

[44] "Riusabilità del software e delle applicazioni informatiche nella pubblica amministrazione", CNIPA, 2004.

[45] Cattaneo F., del Bianco V., Fuggetta A., "Fattori abilitanti al riuso del software nella pubblica amministrazione", CEFRIEL, 2004.

[46] "OpenPA", Internet Website, accessed in June 2007, available on-line via http://www.lugroma.org/openpa.

[47] "Consorzio per il Sistema Informativo (CSI)", Internet Website, accessed in June 2007, available on-line via http://www.csipiemonte.it.

[48] "Piattaforma per la gestione documentale OASI", Internet Website, accessed in June 2007, available on-line via http://www.csipiemonte.it/progetti/oasi.

[49] "Strategie Digitali", Internet Website, accessed in April 2007, available on-line via http://www.strategiedigitali.it.

[50] "FLOSS Piemonte", Internet Website, accessed in April 2007, available on-line via http://floss.piemonte.it.

[51] Fenton N., Pfleeger S. L., "Software metrics: a rigorous and practical approach", PWS Publishing, 1998.

9 APPENDICES

9.1 Goal and Factor Statistical Analysis

The most important factors were ranked in the questionnaire according to an integer scale (0 ≤ factor value ≤ 10): these are the factors that we can reason about with the help of a statistical analysis. There are 37 usable factors in our questionnaire, as shown in Table 9-1.

Table 9-1: List of factors by section

section	*mnemonic*	*factor*
Economic	ROI	ROI
Economic	TCO	TCO
Development	type of licenses	the type of licenses used
Development	tools	the availability of tools for developing modifying customizing OSS products
Development	best practices	the availability of best practices on the specific OSS products
Development	documentation	the availability of technical documentation / user manual
Development	environment	environmental issues
Development	training / guidelines	the availabiliity of training, guidelines, ecc.
Development	user community	the mid / long term existence of a user community
Development	maintainer organization	the mid / long term existence of a maintainer organization / sponsor
Development	short term support	the short term support
Development	reputation of vendor	the reputation of the OSS vendor
Development	distribution channel	the distribution channel
Development	language uniformity	the programming language uniformity
Development	user community that witness quality	the existence of a sufficiently large community of users that can witness its quality
Development	benchmarks / test suites	the existence of benchmarks / test suites that witness for the quality of OSS
Quality	functional requirements	the degree to which an OSS product satisfies / covers functional requirements
Quality	eq reliability	external quality reliability
Quality	eq performance	external quality performance
Quality	eq usability	external quality usability
Quality	eq maintainability	external quality maintainability
Quality	eq portability	external quality portability
Quality	eq reusability	external quality reusability
Quality	iq size	internal quality size
Quality	iq complexity	internal quality complexity
Quality	iq modularity	internal quality modularity
Quality	iq standard architecture	internal quality standard architecture
Quality	iq patterns	internal quality patterns
Quality	iq security	internal quality security
Quality	standard compliance	standard compliance
Quality	self containedness	self containedness
Quality	interoperability	interoperability
Quality	localization	human interface language / localization
Customer	customer satisfaction	customer satisfaction
Customer	interoperability issues	interoperability issues
Customer	law	law conformance
Customer	standard imposed	standard imposed

9.1.1 Distributions

The first analysis of the gathered data we made is solely based on a descriptive statistic such as the mean, which gives a rough idea about the overall preference on trustworthiness factors. However, these results need to be validated through statistical test to assess their statistical significance. Table 9-2 shows the factors means clustered by the class to which the factor belongs, while Table 9-3 shows the factors are ordered by their means (in descending order).

Table 9-2: Factors (1-10) means (*Main set*)

section	*factor*	*mean*
Economic	ROI	6.362
Economic	TCO	6.081
Development	type of licenses	6.803
Development	tools	6.569
Development	best practices	6.181
Development	documentation	7.733
Development	environment	6.836
Development	training / guidelines	4.880
Development	user community	7.284
Development	maintainer organization	5.640
Development	short term support	6.987
Development	reputation of vendor	5.595
Development	distribution channel	3.417
Development	language uniformity	5.806
Development	user community that witness quality	7.054
Development	benchmarks / test suites	5.616
Quality	functional requirements	8.609
Quality	eq reliability	8.082
Quality	eq performance	7.096
Quality	eq usability	7.000
Quality	eq maintainability	7.944
Quality	eq portability	6.310
Quality	eq reusability	6.714
Quality	iq size	3.926
Quality	iq complexity	5.696
Quality	iq modularity	7.456
Quality	iq standard architecture	7.368
Quality	iq patterns	5.882
Quality	iq security	5.818
Quality	standard compliance	7.357
Quality	self containedness	5.986
Quality	interoperability	7.931
Quality	localization	5.986
Customer	customer satisfaction	8.043
Customer	interoperability issues	7.833
Customer	law	7.030
Customer	standard imposed	6.227

Table 9-3: Factors (1-10) means (*Main set*), ordered by mean (descending)

section	*factor*	*mean*
Quality	functional requirements	8.609
Quality	eq reliability	8.082
Customer	customer satisfaction	8.043
Quality	eq maintainability	7.944
Quality	interoperability	7.931
Customer	interoperability issues	7.833
Development	documentation	7.733
Quality	iq modularity	7.456
Quality	iq standard architecture	7.368
Quality	standard compliance	7.357
Development	user community	7.284
Quality	eq performance	7.096
Development	user community that witness quality	7.054
Customer	law	7.030
Quality	eq usability	7.000
Development	short term support	6.987
Development	environment	6.836
Development	type of licenses	6.803
Quality	eq reusability	6.714
Development	tools	6.569
Economic	ROI	6.362
Quality	eq portability	6.310
Customer	standard imposed	6.227
Development	best practices	6.181
Economic	TCO	6.081
Quality	localization	5.986
Quality	self containedness	5.986
Quality	iq patterns	5.882
Quality	iq security	5.818
Development	language uniformity	5.806
Quality	iq complexity	5.696
Development	maintainer organization	5.640
Development	benchmarks / test suites	5.616
Development	reputation of vendor	5.595
Development	training / guidelines	4.880
Quality	iq size	3.926
Development	distribution channel	3.417

Even though the ordering of their means provides an interesting and expressive piece of information, factors cannot be compared directly using their means to find out the actual preference order. There are two reasons for this. First, the importance of the single factors is not measured by an interval or ratio scale, for which the mean is guaranteed to be a fully meaningful central tendency indicator. Second, we need to assess the statistical significance of the ordering, that is, we need to know how "reliable" the ordering between two factors actually is.

The very nature of our questionnaire required that we use ordinal scales, so, to assess the statistical significance of the orderings, we used three well known non parametric tests to establish the order of factors that are appropriate with ordinal scales: the Sign Test [10] (that is actually based on Binomial distribution

[12])[7], the Mann-Whitney Test and the Wilcoxon Test [11][8]. The factors we are interested in those expressed with a preference value (0 ≤ value ≤ 10). There are 37 usable factors expressed with a preference value in the interviews, leading to 666 possible relations between factors to be verified and assessed. We used 0.05 as the statistical significance threshold, as is usually done in empirical software engineering studies.

The study for the *Main set* gives 410 statistically significant ordering relationships; in particular Sign Test gives 288, Mann Whitney 370, Wilcoxon 379. There is obviously a high overlap; the relationships that all the three employed tests rate as statistically significant are 283

The relationships found group the factors in distinct and ordered groups; every group has many relations with other groups (all of the same kind, that is, for example, every factor of group 4 that has a relation with factors of group 3, share the same higher-lower relation, factors in group 4 are higher than factors in group 3), but no relations with members of the same group. It is to be noted that nothing can be said for factors that lie in the same group, that is, no ordering is possible for factors belonging to the same group.

In Table 9-4 the factors are ordered by the groups found in the *Main set*.

[7] A Binomial test performs a test of a simple null hypothesis about the probability of success in a Bernoulli experiment. In this case (Sign test) is used to test the hypothesis that there is "no difference" between the distributions of two random variables.

[8] Mann-Whitney U test and Wilcoxon test are non-parametric test for assessing whether two samples of observations come from the same distribution. The null hypothesis is that the two samples are drawn from a single population, and therefore that their probability distributions are equal. It requires the two samples to be independent, and the observations to be ordinal or continuous measurements.

Table 9-4: Factors (1-10) groups (*Main set*), ordered by group

section	***factor***	***group***
Quality	functional requirements	8
Customer	interoperability issues	7
Quality	interoperability	7
Quality	eq reliability	7
Quality	eq maintainability	7
Customer	customer satisfaction	7
Quality	user community	6
Quality	documentation	6
Quality	standard compliance	6
Development	iq standard architecture	6
Development	iq modularity	6
Development	type of licenses	5
Quality	short term support	5
Quality	user community that witness quality	5
Customer	tools	5
Economic	ROI	5
Development	law	5
Development	eq usability	5
Development	eq performance	5
Development	environment	5
Quality	reputation of vendor	4
Development	eq portability	4
Quality	language uniformity	3
Quality	benchmarks / test suites	3
Quality	best practices	3
Quality	TCO	3
Quality	standard imposed	3
Customer	self containedness	3
Economic	iq patterns	3
Development	iq complexity	3
Development	localization	3
Development	eq reusability	3
Development	maintainer organization	2
Development	training / guidelines	2
Quality	distribution channel	1
Quality	iq size	1
Development	iq security	1

The relations found between factors exhibit a quite good agreement with the relations that could be deduced from mean values; nevertheless some exceptions do exist, as marked in Table 9-5. Some of the deviations are due to an insufficient number of responses, as for the *eq reusability* and *iq security*. The others are important deviations, even though they are not so big, since there is only a one place deviation.

Table 9-5: Factors (1-10) groups and means (*Main set*), ordered by group. Italic marks the factors with an ordering mismatch

section	***factor***	***group***	***mean***
Quality	functional requirements	8	8.609
Quality	eq reliability	7	8.082
Customer	customer satisfaction	7	8.043
Quality	eq maintainability	7	7.944
Quality	interoperability	7	7.931
Customer	interoperability issues	7	7.833
Development	documentation	6	7.733
Quality	iq modularity	6	7.456
Quality	iq standard architecture	6	7.368
Quality	standard compliance	6	7.357
Development	user community	6	7.284
Quality	eq performance	5	7.096
Development	user community that witness quality	5	7.054
Customer	law	5	7.030
Quality	eq usability	5	7.000
Development	short term support	5	6.987
Development	environment	5	6.836
Development	type of licenses	5	6.803
Quality	*eq reusability*	*3*	*6.714*
Development	tools	5	6.569
Economic	ROI	5	6.362
Quality	eq portability	4	6.310
Customer	standard imposed	3	6.227
Development	best practices	3	6.181
Economic	TCO	3	6.081
Quality	localization	3	5.986
Quality	self containedness	3	5.986
Quality	iq patterns	3	5.882
Quality	*iq security*	*1*	*5.818*
Development	language uniformity	3	5.806
Quality	iq complexity	3	5.696
Development	*maintainer organization*	*2*	*5.640*
Development	benchmarks / test suites	3	5.616
Development	*reputation of vendor*	*4*	*5.595*
Development	training / guidelines	2	4.880
Quality	iq size	1	3.926
Development	distribution channel	1	3.417

Table 9-6: Factors (1-10) groups for the *Main set* and the *Whole set*, ordered by group. Italic marks the factors with an ordering mismatch

section	***factor***	***main group***	***whole group***	***main mean***	***whole mean***
Quality	functional requirements	8	8	8.609	8.588
Quality	eq reliability	7	8	8.082	8.194
Customer	customer satisfaction	7	7	8.043	7.716
Quality	eq maintainability	7	7	7.944	7.856
Quality	interoperability	7	7	7.931	7.895
Customer	interoperability issues	7	7	7.833	7.588
Development	documentation	6	7	7.733	7.848
Quality	iq modularity	6	6	7.456	7.457
Quality	iq standard architecture	6	6	7.368	7.424
Quality	standard compliance	6	6	7.357	7.372
Development	user community	6	6	7.284	7.309
Development	environment	5	5	6.836	6.830
Quality	eq performance	5	5	7.096	7.337
Quality	eq usability	5	5	7.000	7.196
Customer	law	5	5	7.030	6.891
Development	tools	5	5	6.569	6.844
Development	user community that witness quality	5	5	7.054	7.204
Development	short term support	5	5	6.987	6.909
Development	type of licenses	5	4	6.803	6.441
Economic	*ROI*	*5*	*3*	*6.362*	*5.722*
Quality	eq portability	4	4	6.310	6.500
Development	*reputation of vendor*	*4*	*2*	*5.595*	*5.643*
Quality	*eq reusability*	*3*	*4*	*6.714*	*7.033*
Quality	localization	3	4	5.986	6.094
Development	language uniformity	3	4	5.806	6.181
Quality	iq complexity	3	3	5.696	5.935
Quality	iq patterns	3	3	5.882	5.870
Quality	self containedness	3	3	5.986	6.319
Customer	standard imposed	3	3	6.227	5.899
Development	best practices	3	3	6.181	6.232
Development	benchmarks / test suites	3	3	5.616	5.677
Economic	TCO	3	2	6.081	5.633
Development	training / guidelines	2	2	4.880	5.081
Development	maintainer organization	2	2	5.640	5.687
Quality	*iq security*	*1*	*3*	*5.818*	*6.214*
Quality	iq size	1	1	3.926	4.163
Development	distribution channel	1	1	3.417	3.438

It can be seen, in Table 9-6, that there is a high accordance between the groups of the *Main set* and the *Whole set*. This is interesting, because the means of the two sets, especially the means related to the Economics factors (*ROI* and *TCO*), varies sensibly. Again, some of the main mismatches are concentrated on the factors that have a very high rate of missed responses. It is also to be noted that, despite the many differences in groups, the differences are always compatible; that is, all the factors can be reordered (inside a group of a given set) to match the ordering given from the groups of a different set.

9.1.2 Associations

Associations among the factors that characterize the interviewee (personal information, education, role, etc.) and the factors that characterize the perception of the trustworthiness of an OSS product have been calculated.

The factors that characterize the interviewees have a nominal scale: a two values {"Yes", "No"} scale, except for type of organization which has three values {"Private", "Public Administration", "No Profit"}. The factors that characterize the choice of OSS have an ordinal/interval scale. For each characterizing factor, for each value of the scale, the interviewees' set has been divided in two subsets: the set with the interviews that have the characterizing factors equal to the scale value, and the set of remaining interviews. For instance, in a test we divide the interviews into the subset originated by No profit organizations and the subset originated by organizations that are not No profit, i.e., that are either Public Administrations or Private.

The pairs of sets have been compared using the Mann-Whitney U test to find out the correlations with statistical significance (as usual, we use a threshold of 0.05).

We considered the factors that characterize the choice of OSS an ordinal scale, as we did for factors comparison; hence we used a non parametric test such as Mann-Whitney U test.

When the factor that characterize the interviewees have only 2 values {"Yes", "No"}, the sets are only constructed for one of the values (the "Yes" value).

The interviews with missing answers concerning one of the two factors considered in every test are eliminated: in other words, the interviews with missing answers are not used to calculate the test nor the means.

The results of the tests, selecting only the results that indicate statistical significance, are shown in Table 9-7. The test are grouped by the type of the characterizing factor: {role, type of organization } (see sections 3.2.1and 3.2.2).

Table 9-7: Associations. "mean (factor = level)" is the mean calculated on the set of interviewees that have factor = level, "mean (factor ≠ level)" is the mean calculated on the set of interviewees that have factor ≠ level

factor	*Level*	*mean factor*	*mean (factor = level)*	*mean (factor ≠ level)*
role developer	Yes	customer satisfaction	7.286	8.548
role developer	Yes	standard compliance	8.033	6.850
role developer	Yes	standard imposed	6.926	5.744
role developer	Yes	user community that witness quality	6.400	7.500
role developer	Yes	type of licenses	7.833	6.049
role project manager	Yes	customer satisfaction	9.308	7.754
role project manager	Yes	user community	8.714	6.950
role project manager	Yes	short term support	8.143	6.721
role upper management	Yes	interoperability issues	7.174	8.143
role upper management	Yes	standard imposed	5.045	6.818
type of organization	No_Profit	ROI	3.167	6.629
type of organization	Private	ROI	6.891	3.923
type of organization	Public	functional requirements	7.333	8.737

From the associations found we extracted only the explainable results: there are other results concerning the role of the interviewee's organization and the education of the interviewees that could not be reasonably explained. The excluded results are shown in Table 9-8.

Table 9-8: Excluded associations. "mean (factor = level)" is the mean calculated on the set of interviewees that have factor = level, "mean (factor ≠ level)" is the mean calculated on the set of interviewees that have factor ≠ level

factor	*level*	*factor*	*mean (factor = level)*	*mean (factor ≠ level)*
education high school	Yes	user community	7.208	10.000
education master	Yes	localization	7.069	5.256
education master	Yes	interoperability issues	8.536	7.386
education master	Yes	ROI	5.167	7.000
education master	Yes	self containedness	6.786	5.452
education master	Yes	documentation	8.586	7.196
education master	Yes	benchmarks / test suites	6.714	4.933
education phd	Yes	eq performance	5.250	7.323
education phd	Yes	eq reusability	2.333	7.444
education phd	Yes	eq usability	5.500	7.188
education phd	Yes	ROI	3.750	6.705
education phd	Yes	standard compliance	6.143	7.492
education phd	Yes	TCO	3.667	6.415
org role customized / configured	Yes	eq maintainability	8.172	7.000
org role customized / configured	Yes	eq portability	6.632	5.000
org role customized / configured	Yes	iq modularity	7.636	6.692
org role customized / configured	Yes	type of licenses	7.250	4.364
org role part of products	Yes	language uniformity	6.408	4.522
org role provide services	Yes	customer satisfaction	8.511	7.087
org role provide services	Yes	tools	7.250	5.208
org role provide services	Yes	type of licenses	7.480	5.190
org role support internal processes	Yes	TCO	5.531	7.160
org role support internal processes	Yes	language uniformity	6.353	4.476
org role support sw development	Yes	eq performance	6.769	7.905
org role support sw development	Yes	eq usability	6.647	7.857
org role support sw development	Yes	TCO	5.558	7.318
org role support sw development	Yes	language uniformity	6.385	4.300
org role development platform	Yes	tools	6.956	5.067
org role development platform	Yes	distribution channel	5.600	1.000
org role development platform	Yes	language uniformity	6.435	4.400
org role target platform	Yes	training / guidelines	5.300	3.867
org role target platform	Yes	tools	7.208	4.600
org role target platform	Yes	user community	7.480	6.133

9.2 Questionnaire and interviews update

9.2.1 The Questionnaire

The new Questionnaire that has been used to conduct the new interviews is a simplified version of the previous Questionnaire. It retains all the closed questions of the previous questionnaire, but most of the open questions have been eliminated. The first round of interviews has allowed us to obtain all of the factors that were deemed important, so it was possible to close most of the open questions mainly to reduce the compilation time of the questionnaire and make it possible to prepare an online version of it. Also, in the first round of interviews, enough qualitative information has been collected, on the contrary quantitative information was still needed to confirm or correct the statistical results that have been found for the first round of interviews; this consideration further justifies the decision to eliminate most of the open questions.

9.2.2 Data collection

The data collection in the second round of interviews has been carried out in the same way as it was in the first round. The interviews have been carried out in a synchronous way, mainly by in person interviews, and rarely by phone.

An online version of the questionnaire is available on QualiPSo website to allow for automated interviews collection. The actual online version of the questionnaire is an evolution of the questionnaire originally published in our intranet; it has been secured and made more usable. The online version was made available with the intention to gather even more data to analyze, even though the online version of the questionnaire has not generated a significant number of interviews so far.

9.2.3 The Sample

151 interviews have been collected in the two rounds of interviews. 103 interviews have been collected in the first round and 48 interviews in the second one.

The total sample retains the same properties of the first round sample, that is, it exhibits a fair distribution considering the nationality, role, organization type, OSS usage, etc. of the interviewees. The most significant frequencies are reported in Figure 9-1 and Figure 9-2. It can be noted that there is a slight increase of interviewees with “type of organization Public” if compared to first round data (see section 4.3.4).

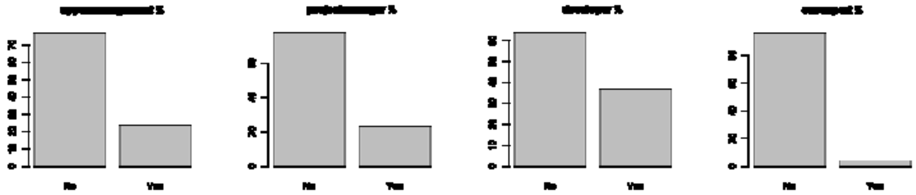


Figure 9-1: Role frequencies

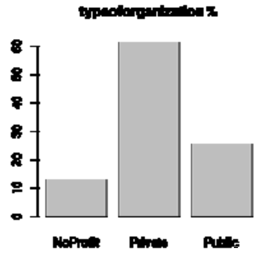


Figure 9-2: Type of organization

9.2.4 Factors analysis

The Factors analysis for the updated sample mainly confirms the findings and the relations previously found for the first round interviews sample. The factors have been divided in 8 groups according to their importance, like for the initial data set. Table 9-9 summarizes the results on the averages and on the importance grouping of factors for both the combined dataset and for only the first data set.

It should be noted that the total number of factors considered here is 37, that is, we are also including two factors that were previously excluded in the main analysis of the results in the first round of interviews (see Section 5). The factors were excluded in the first round of interviews because the data collected for these factors were not sufficient to extract significant results (see Section 3.2.8). In the second round of interviews, the data collected on these two factors were sufficient to extract significant results, but these results should not be compared with the results obtained in the first round, because the latter are not trustable.

It can be noted that the highly differing factors are only a few (4 factors), that is: *ROI* (-3), *reputation of vendor* (-2), *iq modularity* (-2), *iq standard architecture* (-2). The fifth factor that exhibits a strong difference with the first round of interviews, *security* (+4), should not be considered because during the first round of interviews the data available were not sufficient to obtain significant results. The remaining factors maintain the group, or move at most of one group.

Table 9-9: Factors groups and means

section	*factor*	*1°+2° group*	*1° group*	*2° - 1° group differ.*	*1°+2° mean*	*1° mean*
Economic	roi	2	5	-3	5.835	6.362
Economic	tco	2	3	-1	5.750	6.081
Development	type of licenses	4	5	-1	6.821	6.803
Development	tools	4	5	-1	6.810	6.569
Development	best practices	3	3	0	6.355	6.181
Development	documentation	7	6	1	8.068	7.733
Development	environment	4	5	-1	7.025	6.836
Development	training / guidelines	2	2	0	5.692	4.880
Development	user community	6	6	0	7.660	7.284
Development	maintainer organization	2	2	0	5.938	5.640
Development	short term support	5	5	0	7.342	6.987
Development	reputation of vendor	2	4	-2	5.800	5.595
Development	distribution channel	1	1	0	3.438	3.417
Development	language uniformity	3	3	0	6.184	5.806
Development	user community that witness quality	5	5	0	7.552	7.054
Development	benchmarks / test suites	2	3	-1	5.839	5.616
Quality	functional requirements	8	8	0	8.633	8.609
Quality	eq reliability	8	7	1	8.423	8.082
Quality	eq performance	5	5	0	7.455	7.096
Quality	eq usability	5	5	0	7.617	7.000
Quality	eq maintainability	6	7	-1	7.880	7.944
Quality	eq portability	4	4	0	6.705	6.310
Quality	eq reusability	4	3	1	7.067	6.714
Quality	iq size	1	1	0	4.124	3.926
Quality	iq complexity	2	3	-1	5.674	5.696
Quality	iq modularity	4	6	-2	7.328	7.456
Quality	iq standard architecture	4	6	-2	7.139	7.368
Quality	iq patterns	2	3	-1	5.803	5.882
Quality	security	5	1	4	7.644	5.818
Quality	standard compliance	6	6	0	7.563	7.357
Quality	self containedness	2	3	-1	6.123	5.986
Quality	interoperability	7	7	0	8.043	7.931
Quality	localization	3	3	0	6.447	5.986
Customer	customer satisfaction	7	7	0	7.848	8.043
Customer	interoperability issues	7	7	0	7.951	7.833
Customer	law	4	5	-1	6.696	7.030
Customer	standard imposed	2	3	-1	5.878	6.227

Note that ROI and *TCO* belong to the same group: the second round of interviews pushes ROI and TCO as low as group 2 (very low importance).

For illustration purposes only, Table 9-10 provides the ordering of factors according to their importance for the combined data set and the initial data set.

Table 9-10: Factors ordering by group (*1° + 2°* round and *2° round comparison*). Bold values indicates that the ordering is not maintained.

section	*mnemonic*	1°+2°	1°
Quality	functional requirements	8	8
Quality	eq reliability	8	7
Customer	customer satisfaction	7	7
Quality	interoperability	7	7
Customer	interoperability issues	7	7
Development	**documentation**	**7**	**6**
Quality	**eq maintainability**	**6**	**7**
Quality	standard compliance	6	6
Development	user community	6	6
Quality	eq performance	5	5
Quality	eq usability	5	5
Development	user community that witness quality	5	5
Development	short term support	5	5
Quality	**iq security**	**5**	**1**
Quality	**iq modularity**	**4**	**6**
Quality	**iq standard architecture**	**4**	**6**
Development	tools	4	5
Development	environment	4	5
Customer	law	4	5
Development	type of licenses	4	5
Quality	eq portability	4	4
Quality	eq reusability	4	3
Quality	localization	3	3
Development	best practices	3	3
Development	language uniformity	3	3
Economic	**roi**	**2**	**5**
Development	**reputation of vendor**	**2**	**4**
Quality	iq complexity	2	3
Quality	iq patterns	2	3
Quality	self containedness	2	3
Customer	standard imposed	2	3
Economic	tco	2	3
Development	benchmarks / test suites	2	3
Development	training / guidelines	2	2
Development	maintainer organization	2	2
Quality	iq size	1	1
Development	distribution channel	1	1

The ordering found in the first round of interviews is mostly maintained in the second round ranking, as it can be seen in Table 9-10. This is a strong confirmation of the analysis and results found during the first round of interviews.

9.2.5 Associations

Associations among the factors that characterize the interviewee (personal information, education, role, etc.) and the factors that characterize the perception of the trustworthiness of an OSS product have been calculated for the second round of interviewees.

The findings are not promising, that is, they do not confirm the associations found in the first round of interviewees. Specifically, in the first round of interviews, 46 statistically significant associations were found, some of these seemed quite reasonable and were analyzed and commented (13 associations), other associations were not so clear and were left out from the analysis (see section 5.7 and Appendix 9.1.2). In the second round of interviews, 68 statistically significant associations were found, but the intersection of the two associations' sets is composed only of 17 associations. Of these 17 common associations, only 7 associations are in common with the explainable ones from the first round of interviews; we report the associations in Table 9-11.

Table 9-11: Associations

factor	*level*	*mean factor*	*mean (factor = level)*	*mean (factor ≠ level)*
developer	Yes	theexistenceofasufficientlylargecommunityofusersthatcanwitnessitsquality	6.8868	7.9348
projectmanager	Yes	customersatisfaction	9.0000	7.5545
projectmanager	Yes	themidlongtermexistenceofausercommunity	8.4063	7.4464
projectmanager	Yes	theshorttermsupport	8.4375	7.0351
uppermanagement	Yes	standardimposed	4.7500	6.2424
typeoforganization	NoProfit	roireturnofinvestment	3.7333	6.0089
typeoforganization	Private	roireturnofinvestment	6.8750	3.8085

In conclusion, with the sample currently available, nothing reasonable can be said about the statistically significant associations found.

9.3 The questionnaire

9.3.1 Introduction

The purpose of this questionnaire is to elicit information from the users of OSS products about their goals when they use/customize/modify/develop OSS products and about their OSS development processes.

This questionnaire has been developed in the framework of the QualiPSo (Quality Platform for Open Source Software) project, which is a European Union-funded Integrated Project which aims at making a major contribution to the state of the art and practice of Open Source Software. The QualiPSo project started in November 2006 and will last until October 2010. The project brings together over twenty software companies, application solution developers, and research institutions. Its goal is to define and implement technologies, procedures and policies to leverage the Open Source Software development current practices to sound, well-recognized, and established industrial operations.

All information provided by each individual or organization will be treated as confidential. As such, it will not be released in other form than aggregated statistical analyses that will make it impossible to identify the single respondents.

Please do not hesitate to contact us if you need any information or clarification.

9.3.2 Personal information

- Name:
- Role:
- Unit:
- Education:
- Time in the company:
- E-mail:

9.3.3 Company information

- Type of organization (private, no profit, Public Administration, etc.):
- Number of employees:
- Domain(s) (Public Administration, avionics, banking/finance, …):
- Number of employees of the organizational unit:
- Domain(s) (Public Administration, avionics, banking/finance, …) of the organizational unit:

9.3.4 Role of the organization with respect to OSS

- Is the company a producer, user, mixed (user/modifier), value adder (customizer, …) of OSS?
- Choose all that applies:
  1. OSS products are used to support SW development
  2. OSS products are used as part of other product
  3. OSS products are customized/configured
  4. OSS products are used to support the internal process
  5. OSS products are used to provide services to the outside world.
- Is OSS the development platform?
- Is OSS the target/usage platform?

9.3.5 Issues that can be taken into account when deciding whether to adopt OSS

9.3.5.1 Economics

- Do you choose OSS considering (please rank, from 0-irrelevant to 10-essential)
  1. The TCO (Total Cost of Ownership)? E.g., is OSS used because it is less expensive then commercial alternatives?
  2. The ROI (Return On Investment)? E.g., is OSS chosen to reduce effort?
  3. Any other issues related to your business model?

9.3.5.2 License

- What types of licenses do you have in the OSS you deal with?
  - Academic Free License
  - Adaptive Public License (APL)
  - Apache Software License
  - Apple Public Source License
  - Artistic License
  - Attribution Assurance Licenses
  - BSD License
  - Computer Associates Trusted Open Source License
  - Common Development and Distribution License
  - Common Public License
  - CUA Office Public License
  - EU DataGrid Software License
  - Eclipse Public License
  - Educational Community License
  - Eiffel Forum License
  - Entessa Public License
  - Fair License
  - Frameworx License
  - GNU General Public License (GPL)
  - GNU Lesser General Public License (LGPL)
  - Historical Permission Notice and Disclaimer
  - IBM Public License
  - Intel Open Source License

- Jabber Open Source License
- Lucent Public License
- MIT License
- MITRE Collaborative Virtual Workspace License (CVW License)
- Motosoto License
- Mozilla Public License (MPL) 1.0 and 1.1
- NASA Open Source Agreement
- Naumen Public License
- NetHack General Public License
- Nokia Open Source License
- OCLC Research Public License
- Open Group Test Suite License
- Open Software License
- PHP License
- Python License
- Python Software Foundation License
- Qt Public License (QPL)
- RealNetworks Public Source License
- Reciprocal Public License
- Ricoh Source Code Public License
- Sleepycat License
- Sun Industry Standards Source License (SISSL)
- Sun Public License (SPL)
- Sybase Open Watcom Public License
- University of Illinois/NCSA Open Source License
- Vovida Software License v. 1.0
- W3C License
- wxWindows Library License
- X.Net License
- zlib-libpng license
- Zope Public License
- Other

- What should the license allow/restrict to users, developers, modifiers, integrators?
    - Hackers dislike accepting code under it
    - Cannot combine with proprietary and redistribute
    - Cannot combine with GPL'ed code and redistribute
    - Can redistribute binaries without source
    - Apply to everyone who receives the program, without the need for any additional agreements
    - Allow distribution with any other software agreements
    - Allow distribution in any form
    - Grant to distribute the program themselves, including the right to charge money for it
    - Grant the right to distribute modified versions of the program
    - Grant access to the program's source code
    - Grant the right to modify the program

9.3.5.3 *Development Process*

- Do you have a process for selecting OSS to use?
- If so, what is it like?
- Which OSS evaluation methods do you use?
    - QSOS (www.qsos.org)
    - OpenBRR (www.openbrr.org)
    - OSMM - Navica (www.navicasoft.com/pages/osmm.htm)
    - OSMM - Capgemini (www.SeriouslyOpen.org)
    - OpenBQR (http://www.taibi.it/OpenBQR)
- What is the context process in which it is used?
- Do you choose OSS products considering (please rank, from 0-irrelevant to 10-essential)
    1. the type of licenses used?
    2. the availability of tools for developing/modifying/customizing … OSS products?
    3. the availability of best practices on the specific OSS products?
    4. the availability of technical documentation/user manual?
    5. environmental issues (platforms, preferences and needs of personnel, …)?
    6. the availability of training, guidelines, etc.?

7. the mid/long term existence of a user community?
8. the mid/long term existence of a maintainer organization / "sponsor"?
9. the short term support (problem resolution, correction of bugs, etc.)?
10. the reputation of the OSS provider?
11. the programming language uniformity?
12. the existence of a sufficiently large community of users of the OSS software that can witness its quality?
13. the existence of benchmarks, test suites that witness for the quality of OSS?
14. other (please specify)?

- What other characteristics that are not commonly available about OSS development processes would you like to have and use?

9.3.5.4 *Product quality*

- Do you choose OSS products considering (please rank, from 0-irrelevant to 10-essential)

1. the degree to which an OSS product satisfies/covers functional requirements
2. the degree to which other qualities are satisfied, e.g., the qualities of ISO9126
    1. reliability
    2. performance
    3. usability
    4. maintainability
    5. portability
    6. other (e.g., reusability)
3. design and code qualities:
    1. size
    2. complexity
    3. modularity
    4. standard architecture
    5. patterns
    6. other (Please specify)
4. standard compliance
5. self-containedness (the product does not need other "products" to work correctly)
6. the interoperability (data level, formats, etc.)

7. the human interface language / localization of the OSS product

- What other characteristics that are not commonly available about OSS product quality would you like to have and use?

9.3.5.5 *Features supporting the customer requirements*

- What features do you take into account when choosing OSS? (please rank, from 0-irrelevant to 10-essential)

1. Customer satisfaction
2. Interoperability issues
3. Law conformance (e.g., for Public Administrations)
4. Standard imposed
5. other (please specify)

9.3.6 Processes

9.3.6.1 *Trust*

- What are the elements (practices, tools, techniques, etc.) in the process that allow you to trust the quality of the final result?

9.3.6.2 *Quality assurance*

- What are the aspects for verifying quality of he product you use/produce?
- Who is testing the product?
- Which manually test methods are used? (internal/user testing)
- Which automated testing techniques are used?
- How often, how much and what do you test?
- Are new releases scheduled?
- How regularly are releases rolled out?
- Is it planned in which release which :
  - Features will be added?
  - Bugs will be solved?
- How is the work managed in the time of delivering a new release?

9.3.6.3 *General questions*

- Which open source software are used within the company/unit?
- If there is a commercial alternative available, why do you choose OSS?
- Is an OSS product usually used/developed/modified/customized in a single location within the company or at several locations?
- When did the project start?
- Where did the project start?

- Within the company?
- Did the project already have roots/backgrounds (outside of the company), that the company improved?

- How long does it last (approximately)?

9.3.6.4 Roles and responsibilities

- How many people were/are working in the project?
  1. 1-15
  2. 16-25
  3. 26-50
  4. 51-100
  5. 101-500
  6. More than 500
- How much is the turnover? (annual rate of people getting into/leaving the project)
  1. 1%-10%
  2. 11%-20%
  3. 21%-40%
  4. 41%-60%
  5. 61%-80%
  6. 81%-100%
- Please determine:
  - The standard roles:
    1. users (yes/no)
    2. developers (yes/no)
    3. committers (yes/no)
    4. PMC members (yes/no)
    5. other (yes/no)
  - The number of the participants of the project:
    1. users
    2. developers
    3. committers
    4. PMC members
    5. Other
  - The responsibilities:

1. users
2. developers
3. committers
4. PMC members
5. Other

- How can one become a developer, committer, PMC member?
- Is there any community within or outside of the company which makes decisions?
- How are decision processes arranged?
- How do you decide about code modification, giving rights, package releases, etc? (voting, responsibilities, etc.)

9.3.6.5 *Architecture definition*

- How is the technical architecture of the project managed?
- Is it planned before, incremental?
- What are the most important technical requirements?
- Which technologies are used?

9.3.6.6 *Development techniques and practices*

- Which development methodology do you use?
- Can you describe it? (if it is not standard)
- Which practices do you use? (describe it)
  - Test first
  - Unit test
  - Continuous integration
  - Code reviews
  - Other (please specify)
- How do you collect and manage requirements?
- Do you use any coding standards?
- How is the maintenance of the existing code worked out?

9.3.6.7 *Tools used*

- On which operating system is the project implemented?
- Is it running on other OS?
- If yes, on which one(s)?
  - Windows

  - Linux
  - Solaris
  - Other (please specify)
- Which programming language is used for the implementation?
  - Java
  - C++
  - C
  - Visual Basic 6
  - Perl
  - Pyton
  - Other (Please specify)
- On which platform?
  - Windows
  - Linux
  - Solaris
  - Other (please specify)
- Which development tools are used in the project?
  - Eclipse
  - Visual Studio
  - Vi
  - Emacs
  - Other (please specify)
- Do you use any tool developed in house? (yes, no)
- Do you make these tools available to others? (yes, no)
- Do you use other open source or commercial software? (yes/no)

9.3.6.8 *Features to implement*

- Considering the new features; Who:
  1. Makes suggestions for new features? (Is there any mailing list/newsgroups for doing this?)
  2. Is deciding about new features?
  3. Has to implement the new features?
- Is there a time plan
  1. For implementing the features?

2. Which feature should be implemented first? (ranking of features by priorities)
3. How priorities are assigned?

9.3.6.9 Documentation, bug management

- Do you have documentation of the project?
- Who writes the documentation and where? (in the implementation, in a separate documentation, etc.)
- Does the project have a roadmap?
- Is it useful for the developers?
- Which tools are used for bug-tracking?
- If there are several in use, which tool has the highest priority?
- Are the bug-tracking tools specialized for different persons (users, developers, etc), or do they use the same tool for reporting bugs?
- How many bug reports do you get?
- Can the bug-tracking tool be used for other purposes too? (e.g.: making suggestions, looking for tasks to resolve them, etc.)
- How long does it take to solve a bug?
- How are priorities assigned?

9.3.6.10 Version control and people management

- Which version control system is used for the project?
- Is this tool freely available for everybody (user, company, etc.)?
- Who has access to the version control system and which rights?
- Who and how can get more rights and which ones?
- Who can be the owner of a module?
- How are the tasks assigned? Can one choose what to implement?

9.3.6.11 Business model

- Are developers employee?
- Which advantages/disadvantages, benefits has the developer for contributing?
- What is the goal of the project?
- Does the company sell this product?
- Are there any additional services (e.g. courses, support, extensions, etc)?
- If yes, which one(s)?

9.3.6.12 *Workflows of the processes identified*

- Please describe the following processes:

1. Development techniques
2. Release development
3. Testing
4. Quality assurance