Sponsored through Framework Programme Sixth (Call 5) by

Document Information

Version: **1.0**
Date : Mar 3, 08
Pages : 179

Owning Partner:
IESE

Author(s):
Marcus Ciolkowski, Liliana Guzmán, Martín Soto, Flora Kamseu, Dominique Archambeau

Reviewer(s):
Naji Habra, Kirsten Haaland

To:
European Commission

Purpose of distribution:
Final version

The QualOSS Consortium consists of: CETIC (BE), Facultés Universitaires Notre Dame de la Paix à Namur (BE), Universidad Rey Juan Carlos (ES), Fraunhofer IESE (DE), ZEA Partners (BE), MERIT (NL), AdaCore (FR), PEPITe (BE)

Printed
on 03/03/08 at 12:56:58 PM

Status:

[ ] Draft
[ ] To be reviewed
[ ] Proposal
[X] Final/Released

Confidentiality:

[X] Public - Intended for public use
[ ] Restricted - Intended for QualOSS consortium only
[ ] Confidential - Intended for individual partner only

Deliverable ID: **D1.6**

Title:

QualOSS D1.6

(Validation of Calibration and User Manual)

A deliverable of Task 1.6

Deliverable: D1.6

Title: QualOSS D1.6

Executive Summary:
This document describes the work done and the results obtained in task 1.6 (Validation of Calibration and User Manual). With the goal of evaluating the appropriateness of the definition of the QualOSS model from the F/OSS evaluators' viewpoint, we designed and conducted several interviews with partners of the QualOSS consortium and with practitioners who are not part of the QualOSS project. Based on the results, we identified quality characteristics to refine and a list of possible scenarios for planning and scoping the case studies of Workpackage 5. Additionally, a new project has been evaluated using the QualOSS model first released in deliverable D1.5 developed during the previous stages of the project. The purpose of this later subtask was to show that the QualOSS model can successfully be applied to different domains from those considered in Deliverable D1.5. Furthermore, based on the available projects and the extracted metrics obtained during task 1.5, recommendations are discussed and analysed in order to help structure and organize the received raw metrics data files into convenient databases being well suited to carry out later deeper analysis using statistical tools on the one hand, and more advanced data-mining methods on the other hand.

Further work is still required. In particular, the following aspects need to be further evaluated: (1) the completeness and relevance, (2) the usefulness and usability, and (3) the validity and reliability (accuracy) of the QualOSS model. However these issues can only be addressed once the QualOSS tool is implemented. Workpackage 5, which is responsible for planning and conducting case studies, will be directed at this evaluation goals.

The deliverable is structured as follows:

Section 1 presents an introduction to the goals and the approach taken to produce this deliverable.

Section 2 introduces the selected approach for assessing the appropriateness of the QualOSS model, that is, the planning of structured interviews based on the design of a questionnaire.

Section 3 summarizes the results of the conducted interviews in terms of candidate quality characteristics to refine and then to deduce scenarios for case studies.

Section 4 resumes the evaluation of an additional project based on the QualOSS model first release. This section establishes that the QualOSS model can successfully be applied for evaluating projects of different domains from the considered in Deliverable 1.5.

Section 5 discusses recommendations in order to help structuring and organizing the received raw metrics data files into convenient databases being well suited to carry out later deeper analyses using statistical tools, but also more advanced data-mining methods.

Finally, Section 6 summarizes the result of this deliverable and of Task 1.6.

The Appendix contains the design of the survey (full version), the results of each interview and the evaluation of the additional project.

CHANGE LOG

Ver.	*Date*	*Author*	*Description*
0.1	15.08.07	Marcus Ciolkowski	Initial proposal for structure
0.2	05.12.07	Liliana Guzmán	Evaluation goals
0.3	12.12.07	Marcus Ciolkowski	Feedback
0.4	13.13.07	Liliana Guzmán	Rework of section 2.
0.5	13.02.08	Liliana Guzmán	Interview results
0.6	14.02.08	Liliana Guzmán	Executive summary. Conclusions.
0.7	14.02.08	Liliana Guzmán	Rework of section 3 and 6.
0.8	14.02.08	Martín Soto	Add section 4 and proofread whole document.
0.9	25.02.08	Liliana Guzmán	Rework according to first review by Naji Habra.
0.10	29.02.08	Liliana Guzmán	Rework according to the second review by Kirsten Haaland.
1.0	03.03.08	Jean-Christophe Deprez	Sanity Check of front page info and format plus exec summary and change log.

TABLE OF CONTENTS

1 Introduction

1.1 Motivation

The strategic objective of the QualOSS project is to enhance the competitive position of the European software industry by providing methodologies and tools for improving their productivity and the quality of their software products. To achieve this objective, QualOSS notes that many organizations integrate Free / Libre Open Source Software (F/OSS) in their systems, hence QualOSS aims at facilitating the selection of the most adequate F/OSS. In particular, QualOSS focuses on assessing the evolvability and robustness of F/OSS projects.

This higher competitiveness is to be addressed by providing a reliable assessment method of open source software, in order to integrate them into industrial software. This will ease the integration of high quality level open source components, and increase the productivity.

To achieve this goal, QualOSS proposes to build a high level methodology to benchmark the quality of open source software, in order to ease the strategic decision of integrating adequate F/OSS components into software systems. Therefore, one of the main outcomes of the QualOSS project is to deliver an assessment methodology for gauging the evolvability and robustness of open source software.

This first Workpackage (WP1) performs requirements analysis through prototyping, while the other scientific Workpackages (WP2-4) improve on the functional prototype build in WP1. The first three tasks of WP1 (T1.1, T1.2 and T1.3) perform requirements analysis while the remaining three tasks (T1.4, T1.5, and T1.6) build the functional prototype and validate the approach. The goal of this deliverable´, D1.6, is to calibrate the quality model defined in D1.3; that is, to define interpretations where possible, on the basis of the measurements taken in task 1.4.

1.2 Goal

The key result of task 1.6 is the validation of the calibrated QualOSS model, as created by Task 1.5. More specifically, the goals of task 1.6 are to:

- Select a new arbitrary F/OSS project that satisfies the minimum properties needed for our quality model, collect and process raw data for this new F/OSS project and verify that the calibrated QualOSS model still works on the new F/OSS project.
- Verify that the user manual accompanying our quality model is understood by potential users of our quality model.
- Discuss the problems and limitations of the QualOSS quality model. Given the positive results of our additional project evaluation (see Section 4) we consider this part fulfilled by the extensive discussion presented in Section 6 of Deliverable 1.5.

1.3 Strategy For the whole Workpackage 1 (reminder)

The main objective of WP1 is to perform requirement analysis through prototyping. Previous to task 1.5, there existed a quality model; that is, a set of quality characteristics with associated metrics and corresponding measuring tools.

The outcome of prototyping in WP1 serves in performing a thorough requirement analysis in order to test our approaches and to well formulate our requirements for the remainder of the project. It also helps identify promising metrics and tools to integrate in our final QualOSS platform. A first prototype schema for the QualOSS repository also emanates from WP1, in particular from task 1.4. If our prototype quality model constructed on basic metrics and the calibration exercise yield interesting results directly usable and transferable to our QualOSS platform then that is an extra benefit.

The tasks of Workpackage 1 can be grouped as follows: (1) Definition of goals for the QualOSS method, (2) definition of quality models that support these goals, and (3) evaluation and calibration of the quality models.

1. Definition of goals to be supported by the QualOSS method is addressed in task 1.2. Thereby, the approach is to first define and elicit usage scenarios for OSS components, and to define evolvability/ robustness based on these scenarios and on related work in quality modelling and assessment of OSS projects.

2. Definition of QualOSS quality model is addressed in task 1.3. The definition was done top-down as well as bottom-up. The top-down part is addressed by selecting and defining models suitable to meet the previously defined goals, based on a survey of available models as well as of decision makers in industry. This includes existing assessment methods for F/OSS projects, relevant quality models (such as ISO 9126), and on insights from related projects on F/OSS evaluation, such as FLOSSmetrics. In addition, the definition also takes into account available data and tools, as elicited in task 1.1. which shows the inputs for task 1.3. In particular, the usage scenarios' goals and requirements were partially elicited through stakeholders' interviews.

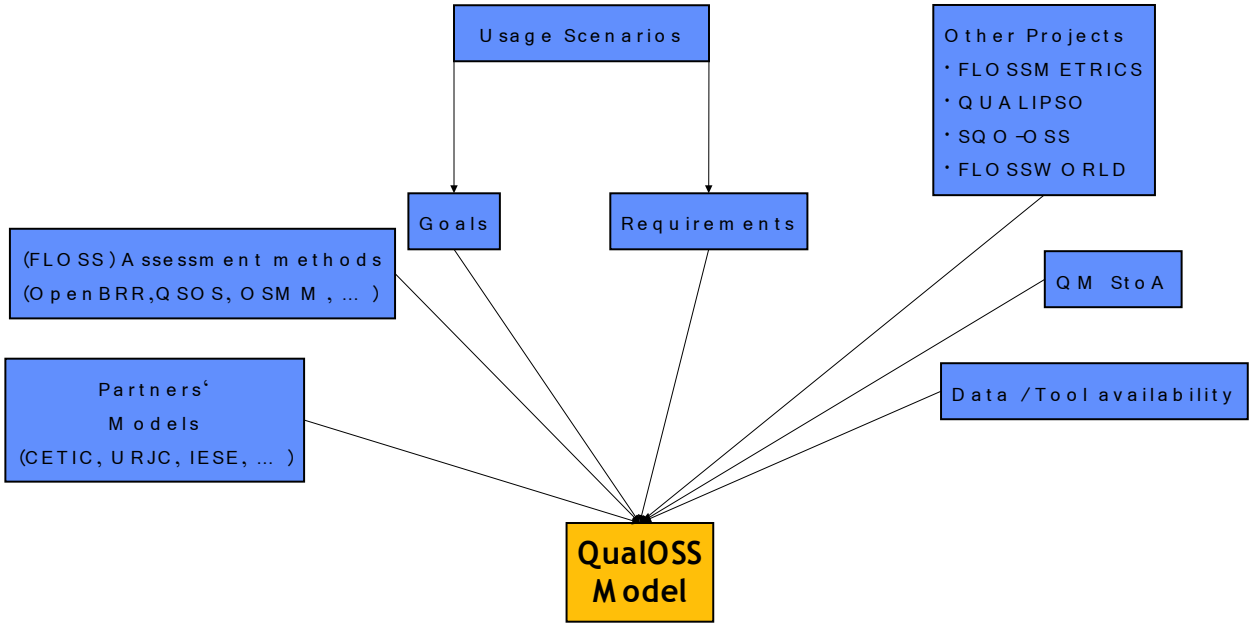


Figure 1: Input Sources for the QualOSS model (D 1.3)

3. Evaluation and calibration of the quality models are addressed in tasks 1.4 to 1.6. Thereby, task 1.4 implements a prototype and repository for data extraction, and uses this prototype to process a set of reference projects. Workpackage 2 will build an advanced set of tools based on the experience gathered in task 1.4. Definition of the interpretation and calibration of the quality models is addressed in task 1.5. More precisely, task 1.5 examines the applicability of the quality models w.r.t their metrics and tries to find patterns and dependencies (through data mining) in the data that can be used as input to improve the quality model. Task 1.6 validates the quality model through interviews and by measurement of additional projects. This includes, for example, evaluating the definition and prioritization of quality characteristics from stakeholders' viewpoint. Workpackages 4 and 5 will pick up on the results of tasks 1.5 and 1.6 by creating advanced quality models and extensively evaluating them.

It is important to note that work in task 1.2 and 1.3 made it clear that we need to restrict D1.2 to definition of robustness and evolvability characteristics. In terms of the goal-question-metric (GQM) paradigm's terminology, these are the measurement goals and questions. The GQM metrics; that is, the definition of appropriate metrics and identification of measurement tools, is part of D1.3. In addition, as product and community aspects need to be considered, and as process maturity is intrinsic to assessing a community, we decided that part of task 1.3 will be to develop an assessment method. The vision of the QualOSS quality model is that all stakeholders use the same definition and metrics to measure robustness and evolvability.

What may change depending of the stakeholder's situation, however, is the "priority" of the quality characteristics w.r.t. stakeholder goal. For example, stability of a product is measured in the same way for all *products*; however, if it is to be used as desktop tool or as part of an external service the company offers, the stability is of different importance to the stakeholder. For this reason, we decided to elicit usage scenarios for F/OSS components. These usage scenarios will later be used to define an initial weighting of the different quality characteristics. The definition of quality characteristics will be independent of the scenario. The challenges that need to be addressed in the QualOSS quality model are missing or inconsistent data; for example.

1.4 Approach

This section describes the approach we took to achieve the goals of Deliverable 1.6.

The goals of D1.6 can be summarized as validating the QualOSS model documentation (User manual) regarding the following questions:

- Model definition understandability: are the characteristics defined so that they are understandable to prospective users?
- Model criteria relevance: are the criteria defined for evaluation relevant to prospective users?
- Model criteria completeness: are there missing criteria which are relevant from prospective users' viewpoint?

In addition, task 1.6 shall deliver an initial weighting scheme for the QualOSS model, and gather stakeholders' input for focusing work in the remainder of the project: (1) their preference on scenarios for case studies (WP5), and their preference of quality characteristics to refine.

1.5 Structure of the Deliverable

The rest of the deliverable is organized as follows: Section 2 presents the design of a structured interview corresponding to the selected approach for evaluating the appropriateness of the QualOSS model (ie. the three questions above). Section 3 summarises the results of the conducted interviews in terms of candidate quality characteristics to refine and scenarios for case studies. Section 4 discusses the evaluation of an additional project based on the QualOSS model first release. Section 5 discusses approaches and recommendations in order to help structure and organize the received raw metrics data files into convenient databases being well suited to carry out later deeper analyses using statistical tools as well as more advanced Data Mining methods. Finally, Section 6 summarizes the result of this deliverable and of Task 1.6

Keywords: Free / Libre Open Source Software, quality modelling, process assessment, project assessment, product assessment, evolvability, robustness

2 Evaluation Goals and Chosen Approach

This section presents the goals and the approach used to evaluate the appropriateness of the QualOSS model for the stakeholders; that is, for the evaluators of F/OSS components. Task 1.6 is concerned with evaluating whether the QualOSS model delivers useful and reliable information to the stakeholders. In order to achieve this purpose, the initial plan for validating the QualOSS model (Deliverable 1.2) defined three potential evaluation goals (EG):

- **EG1:** Evaluate the definition of the quality model (i.e., the quality characteristic definition and prioritization) with the stakeholders.
- **EG2**: Evaluate the usefulness and usability of the QualOSS model. This goal addresses the question to which degree the user believes that the QualOSS provides support for an effective evaluation of the F/OSS components and to which degree using the QualOSS model/tool is free of effort.
- **EG3:** Evaluate the validity and reliability (accuracy) of QualOSS model; that is, the degree to which the results of the QualOSS evaluation reflect the users' intuition and perception of F/OSS components.

The first evaluation goal EG1 has been covered by conducting structured interviews to examine the understandability and completeness of the model definition. The chosen approach also evaluates the relevance of the model´s quality characteristics.
The second and third evaluation goals have been postponed. In part, because the effort for investigating them in task 1.6 would had been too high, but mainly because both imply the availability of the QualOSS tools (which are not included in the prototypical phase) or a ready-to-use model. EG2 and EG3 for the QualOSS model can be addressed in many ways, for example, by reflecting QualOSS evaluations of F/OSS components with experienced users, or by asking experienced users to rank quality characteristics of F/OSS components they are familiar with, and correlating their evaluation with the QualOSS metrics and interpretation. However, EG2 and EG3 for the QualOSS tool and platform can only be addressed once the QualOSS tools are implemented. Workpackage 5, which is responsible for planning and conducting case studies, will address EG2 and EG3.

2.1 Goals

Our primary goal for task 1.6 (EG1) was then to evaluate the definition and the prioritization of the quality characteristics and sub-characteristics of the QualOSS model, compared to the perception and intuition of the evaluators of F/OSS components (see). More precisely, this means to assess the QualOSS model and its sub-characteristics with respect to three aspects:

- **Understandability** – Are the defined quality characteristics understandable/meaningful to the users?
- **Completeness** – Are relevant characteristics missing from the user's point of view?
- **Relevance** - Are the defined quality characteristics relevant for the evaluation of F/OSS components from the user's perspective?

Table 1: **Evaluation Goal 1**. Evaluate the definition of the quality model (i.e., the quality characteristic definition and prioritization) with the stakeholders.

Goal	Validity of the QualOSS model´s definition.
Definition	Evaluate the definition and the prioritization of QualOSS model with stakeholders.
Object	QualOSS model definition
Purpose	Characterize
Quality focus	Validity of the QualOSS model (i.e of the quality characteristic definition and prioritization)

	compared to the perception and intuition of F/OSS evaluators. That means, the understandability, completeness and relevance of the quality characteristics and sub-characteristics.
Viewpoint	F/OSS evaluator

A secondary goal for task 1.6 was to understand how the usage scenarios influence the user´s perception of the relevance of the quality characteristics. Usage scenarios capture different types of settings in which an organization will use F/OSS components. Thus, different usage scenarios may imply different evaluation goals for the company and may therefore change the relevance of different quality characteristics. For example, the quality of developer documentation is crucial if the company plans to extend functionality (e.g., building plugins for Eclipse), but not relevant if the F/OSS component is to be used as-is (e.g., when using Eclipse as development environment only).

2.2 Approach: Structured Interviews

To evaluate the QualOSS model definition, we had two alternatives: questionnaires or interviews. We wanted to examine the understandability, completeness and relevance of the QualOSS model characteristics from the F/OSS evaluator's perspective, and to gain a better understanding of their preferences regarding which quality characteristics to refine further. Therefore, we chose to develop structured interviews, because they allow to clarify and to delimit open-ended questions. The following section describes the survey design used for conducting the interviews.

2.2.1 Conceptual Model

Table 2 specifies the objects of interest of the survey, namely, the organizational context, the model definition and the prioritization of the quality characteristics.

In addition, the survey asks for the F/OSS evaluator's preference on scenarios for case studies, with the purpose of better focusing the work done in the subsequent project tasks and QualOSS model evaluation.

Table 2: Conceptual model

Object	***Description***
Organizational context	Company Granularity of F/OSS components (*usage scenario*) Intended F/OSS usage
Definition of QualOSS model	*Understandability* of the quality characteristics and sub-characteristics Completeness of the QualOSS model
Prioritization of QualOSS model	*Relevance* of the quality characteristic and sub-characteristics from the evaluator's viewpoint
Case study preference	Preference and requests for evaluation support by QualOSS

- *The organizational context* characterizes how the respondent's organization uses F/OSS components. The purpose is to classify organizations, in order to be able to analyse how context factors (i.e., usage scenarios for F/OSS components) influence the user's perception of the relevance of the quality characteristics.

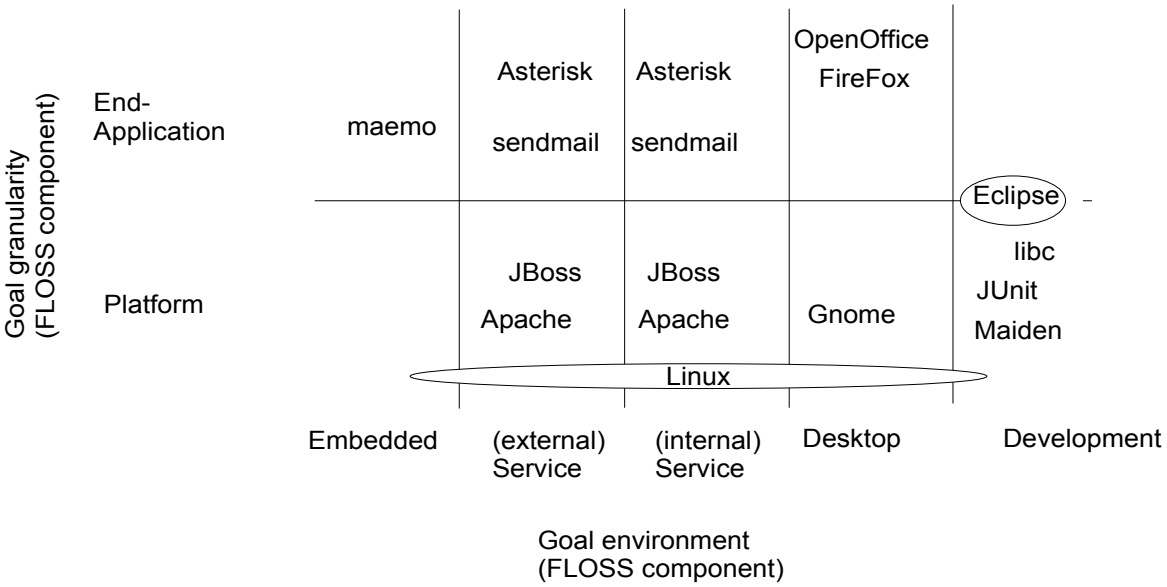


Figure 2: Structure of F/OSS usage scenarios

In Deliverable 1.2, usage scenarios have been classified along the granularity of the F/OSS component and the goal environment. The granularity of a F/OSS component determines whether it is used as is (i.e., as end-application), or whether it is used to build an application (i.e., as platform). The goal environment for F/OSS component describes in which context the F/OSS component is intended to be used (See Figure 2). Scenarios have been also characterized by the intended F/OSS usage, type of F/OSS component and characteristics of the products that integrate F/OSS components. The impact analysis of all possible combinations of the variable defined above to describe a usage scenario, requires a large sample. Therefore, we limited our focus to the granularity of F/OSS components and the intended F/OSS usage.

- *The definition of the QualOSS* model addresses how meaningful (i.e., understandable/intuitive) the definition of the quality characteristics is for the respondent. Completeness addresses whether relevant characteristics are missing from the respondent's viewpoint. The purpose is to prioritize quality characteristics for future refinement in QualOSS, as well as omissions of important quality characteristics.

- *The prioritization of the QualOSS mode*l focus on the relevance that the respondent assigns to quality characteristics proposed in the QualOSS model as well as on the identified missing characteristics. The purpose is to understand how the importance of the quality characteristics changes according to the granularity of the F/OSS components.

2.2.2 Target Population and Sample

The survey is aimed at organizations that are currently using F/OSS in any way. For example, organizations that are currently using F/OSS as part of their information infrastructure and organizations that are developing and offering F/OSS products or services to the public. We selected a convenience sample, because tasks 1.6 is concerned only with the initial validation of the QualOSS model and the user manual (i.e. interpretation model), and because of the F/OSS evaluator's availability. We interviewed 6 industrial partners, most of whom were responsible for IT in their organizations at the time they were interviewed. The sample included four different domains: Public administration, E-government, Research Centre, and Development for the public sector.

It is worth noting here that using a convenience sample may result in biased conclusions and restricts the validity of the results only to the interviewed organizations. For example, the results may be biased by organizations interested only in a specific subset of usage scenarios. Thus, the results show only candidate

quality characteristics to be refined. To cover this issue, the discussion about the quality characteristic refinement considers also some practitioners from the F/OSS community.

2.2.3 Questionnaire Design

We designed the questionnaire according to the evaluation goals, the survey conceptual model and the QualOSS model proposed in Deliverable 1.3. The current QualOSS model focuses on evaluating the evolvability and robustness of F/OSS components. Each of these characteristics have been decomposed into product and community quality characteristics and sub-characteristics.

The questionnaire design includes four parts: The first part focus on characterizing the respondent and his organization. The second part introduces the quality model for evaluating evolvability, presents the definition of the quality characteristics according to the model hierarchy, and asks the respondent to rate how meaningful (understandable) and relevant the characteristic are from his viewpoint, when assessing F/OSS components. Rating the understandability was done by ordering the quality characteristic definition in the following categories: completely meaningful, mostly meaningful, mostly meaningless and completely meaningless. Rating the relevance was done by assigning numbers between 1 and 10. This part also asks for relevant missing characteristics for the evaluation of the F/OSS components and their relevance; that is, for the completeness of the QualOSS model. The third part covers the quality model for evaluating robustness of F/OSS components and has the same structure as the second part. The last part tries to elicit potential usage scenarios for the QualOSS platform, that is, scenarios for which stakeholders would like to have support when evaluating F/OSS components.

In total, the current QualOSS model has defined 40 quality characteristics for evolvability and 23 for robustness. The survey covers all these quality characteristics and has 207 questions (See Table 3), including 18 open questions for examining the model completeness. The complexity of the survey resides in the amount of questions and the dependency between the quality characteristics. The quality model's hierarchy implies that the respondent should evaluate each quality sub-characteristic in the context of a specific quality characteristics.

Table 3: Number of quality characteristics covered by the survey.

Main quality characteristics		**Number of characteristics**	**Number of sub-characteristics**	**Number of questions**
Evolvability	Product Evolvability	6	16	72
	Community evolvability	5	13	59
Robustness	Product Robustness	4	10	46
	Community Robustness	3	6	30
Total		18	45	207

2.2.4 Validity considerations

The validity of a study indicates the strength of its conclusions. Validity is usually classified into internal, external and construct validity. *Internal validity* determines the degree to which conclusions about cause-effect-relations are likely to be true, by means of the research design, research setting, measurements used, etc. For instance, internal validity deals with questions such as: "Does the previous experience and knowledge of the respondents bias the result?", "Does the survey design (e.g. the question phrasing, the order of questions, or the questionnaire extension) influence the respondent's answer?", or "Does the interviewer expectations biases the respondent´s answers?" *External validity* refers to the ability to generalize the results to other settings. *Construct validity* identifies if the study measures what is intended to measure. For a complete discussion of validity, please refer to [1].

In our case, the main concerns related to validity are:

- **Internal validity.** Internal validity addresses if the design of the study enables to obtain valid conclusions. In a survey the control is usually low; for example, it is impossible to determine if the respondent answered truthfully. The following threats have been considered:
  - *History.* History refers to external unexpected events (e.g. recent experiences or the presence of supervisors) which may influence subjects during the application of the questionnaire. To reduce the influence of external events, we chose an approach based on a structured interview.
  - *Maturation*. The application of the whole questionnaire demands half a day. Thus, the respondent's answer may change during the interview due to fatigue. To reduce this threat, we conducted internal interviews (i.e., with partners of the QualOSS consortium), using the full version of the survey, and external interviews (i.e., with respondents who were not part of the QualOSS project), using a shortened version concerning only the first two abstraction levels of the QualOSS model for evaluating evolvability and/or robustness.
  - *Experimenter bias*. Experimenter bias is concerned with the impartiality of the person in direct contact with the subjects. In this case, the interviewer's attitude can predetermine the respondent's answers. This threat has been covered by conducting structured-interviews and by a balance between close and open questions.
  - *Non response.* As explained above (See section 2.2.2) we used a convenience sample, which implied the absence of the opinion of one or more population segments. Although the selected sample range covers different domains and company sizes, we don't expect to have a representative sample.
- **External validity.** External validity is directly related to the representativeness of the sample for the target population. We used a convenience sample in order to accommodate the F/OSS evaluators' time restrictions. Therefore, the validity of the results is restricted to the interviewed organizations and it shows only candidate quality characteristics to refine. To deal with this issue and to prioritize the scenarios for the case studies (WP5), we discussed the results with practitioners from the F/OSS community.
- **Construct validity.** This assess whether the questionnaire measures properly the understandability, completeness and relevance of the quality characteristics. In this case, the construct validity is concerned with the ambiguity of the questions and the conceptual model. Both have been addressed by (1) assessing the quality characteristics according to the model dependencies, that is, according to its specific context (e.g. product evolvability, code document usefulness), (2) providing the definition of each quality characteristic at the beginning of its evaluation, (3) using scales recommended in the literature to measure relevance [2, 3], and (4) conducting a peer review and a pilot test.

Because of the available funds, and consequently available time, we were not able to conduct an extended validation. It is also important to remark that interviews offer the possibility to clarify ambiguities in the questions during the application of the survey.

2.2.5 Survey implementation

Since applying the full version of the survey lasts about half a day per interview and given the limited availability of F/OSS evaluators, we planned to use two versions of the survey. We conducted internal interviews (i.e., with partners of the QualOSS consortium), using the full version of the survey, and external interviews (i.e., with respondents who were not part of the QualOSS project), using a shortened version concerning only the first two abstraction levels of the QualOSS model for evaluating evolvability and/or robustness. The questionnaire could be found in Appendix A.

3 Interview Results

This section summarizes the results of the interviews. In total, we conducted interviews in six industrial partner organizations from various domains: public administration, e-government, research, and development for the public sector.

Due to the limited time available to our industrial partners, 4 of 6 interviews used a shortened version of the survey (See Table 4). Therefore, the analysis of the results covers only the first two abstraction levels of the QualOSS model. Because of time constraints, and the difficulty to find suitable F/OSS evaluators, it was not possible to conduct additional interviews in this prototypical phase in order to get a larger coverage of the model. A complete evaluation of the QualOSS model will be addressed by Workpackage 5.

Table 4: Interviews scope.

Organization ID	Interview scope					Granularity level	
	Survey type	Coverage of the QualOSS Model				End application	Platform level
		Product evolvability	Community evolvability	Product robustness	Community robustness		
1	Short version	X	X	X	X	X	X
2	Short version	-	X	-	X	X	X
3	Short version	X	X	X	X	X	X
4	Full version	X	X	X	-	X	X
5	Full version	X	X	X	X	X	X
6	Short version	X	-	-	-	X	X
Subtotal		***5***	***5***	***4***	***4***	***6***	***6***

The main goals of this analysis are the identification of candidate quality characteristics to refine in the next phase and the F/OSS evaluators' preferences on scenarios for future case studies (Workpackage 5). Consequently, the subsections described below address the following questions:

- Definition of the QualOSS model – Are the defined quality characteristics understandable to F/OSS evaluators? From the point of view of the F/OSS evaluators, have relevant quality characteristics been omitted?
- Prioritization of the QualOSS model – Are the defined quality characteristics relevant to F/OSS evaluators and if so, in which degree?
- Scenarios for case studies – Which are the F/OSS evaluators' preferences on scenarios for case studies?

3.1 Definition of the QualOSS model

Table 5 and the diagrams in Figure 3 summarize the perception of the respondents regarding how understandable the defined quality characteristics of the QualOSS model are. Table 5 shows the percentage of F/OSS evaluators, who assess the definition of the quality characteristics as meaningful. Figure 3 shows

the number of respondents who asses the definition of the quality characteristics as meaningful, meaningless and the number of non responses. This analysis has reconsidered the answer categories “completely meaningful” and “mostly meaningful” as meaningful, and the categories “completely meaningless” and “mostly meaningless” as meaningless.

In general, the respondents appraised the QualOSS model as *understandable*. However, they pointed out specific ambiguities in some specific quality characteristic definitions. Regarding the quality characteristics for evolvability, the respondents made the following comments:

1. The difference between the usefulness of code and usefulness of user documentation is unclear.
2. The definition of interoperability is a self-referenced definition, it does not comply with the IEEE 610 standard.
3. The meaning of the word “standard” should be specified

With respect to robustness, the respondents emphasized the following issues:

1. It is unclear what “correct” means in the context of product robustness.
2. The maturity is not always a long-term issue.
3. In the case of community robustness, the definition of *maturity of the security* process should cover explicitly preventive and reactive actions. The definition of *maturity of reliability* should specify the meaning of “critical” and should precise the considered perspective (developer or user point of view).

Table 5: Summary of meaningful of the definition of QualOSS model

	Meaningful
Product evolvability	**100.0%**
Usefulness of code documentatior	60.0%
Usefulness of User Documentatior	100.0%
Maintainability	60.0%
Portability	100.0%
Interoperability	80.0%
Compliance to standards	40.0%
Community evolvability	**80.0%**
Product Adoption	100.0%
Developer community liveliness	100.0%
Process maturity	100.0%
Support availability	100.0%
Product robustness	**75.0%**
Reliability	50.0%
Maturity	100.0%
Security	75.0%
Community robustness	**50.0%**
Maturity of security process	75.0%
Maturity of reliability process	75.0%

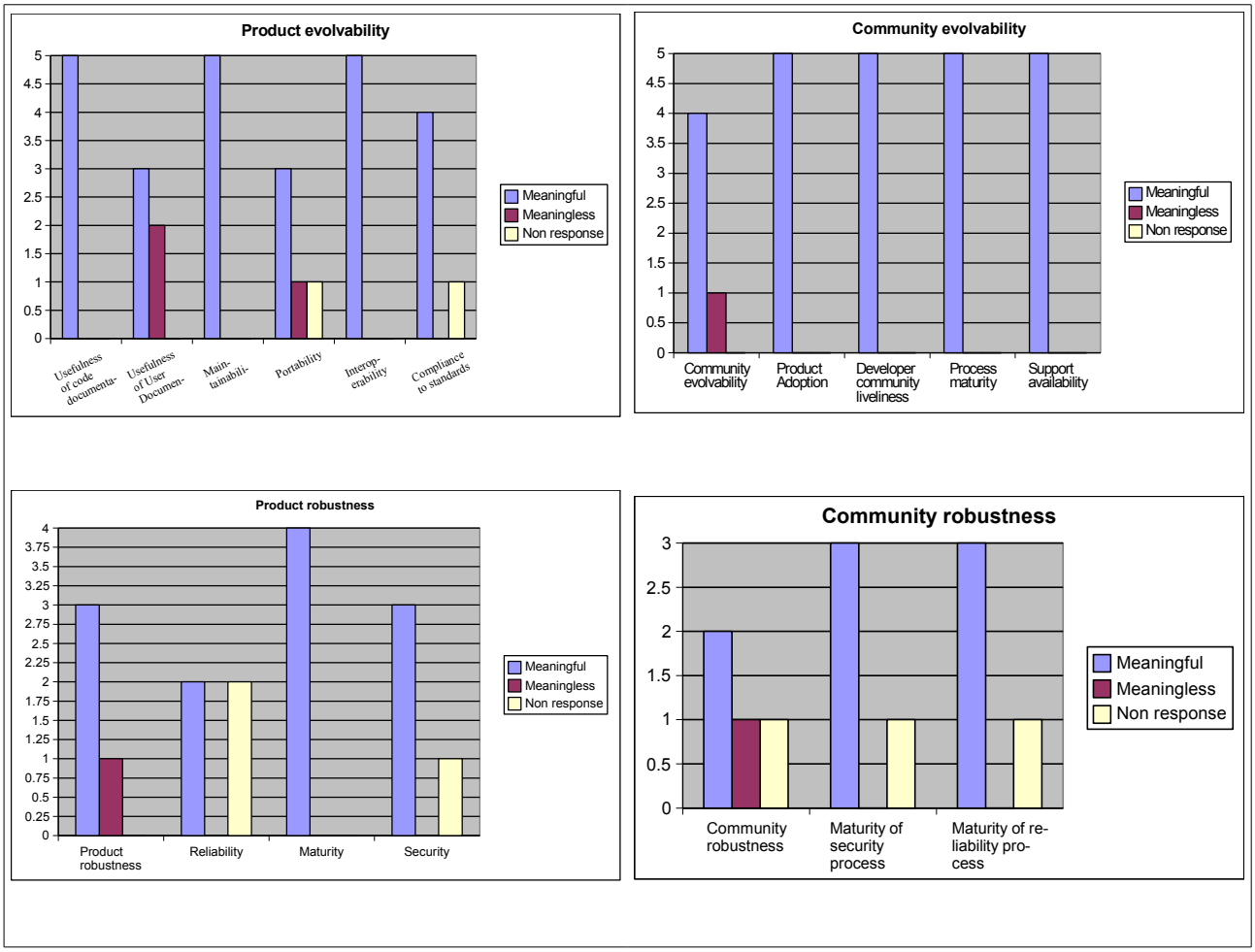


Figure 3: Meaningfulness of the definition of the QualOSS model.

Regarding the completeness of the QualOSS model, different respondents suggested several extra quality sub-characteristics to be considered in the model, but their suggestions did not coincide. The recommended quality characteristics are:

- Product evolvability
  - Availability of information in the Internet: For example, the existence of blogs, mailing lists or forums.
  - Community capability and experience: Experience of the community performing maintenance in general and in a specific F/OSS products.
  - "Evol-Centricity" – (1) The degree to which product evolution and maintenance are treated as central activities in the development process by the community and (2) the degree to which they are reflected in the tools selected by the F/OSS endeavour.
- Community evolvability
  - Commitment of large companies.
  - Taste of innovation.

- Balance of power: The degree to which the power between the different actors of the community is well-balanced.
- Community interoperability – The degree to which a community is able to collaborate with other F/OSS endeavours.

- Community robustness
  - Licensing.

3.2 Prioritization of the QualOSS model

Figure 4 shows the respondents' opinions about the relevance of the quality characteristics according to the granularity level, that is, whether the F/OSS component is used at the end-application level or at the platform level. The results do not show a significant difference between product and community quality characteristics of evolvability and robustness, nor do they establish a significant difference between quality characteristic relevance for F/OSS components at end-application level and at platform level.

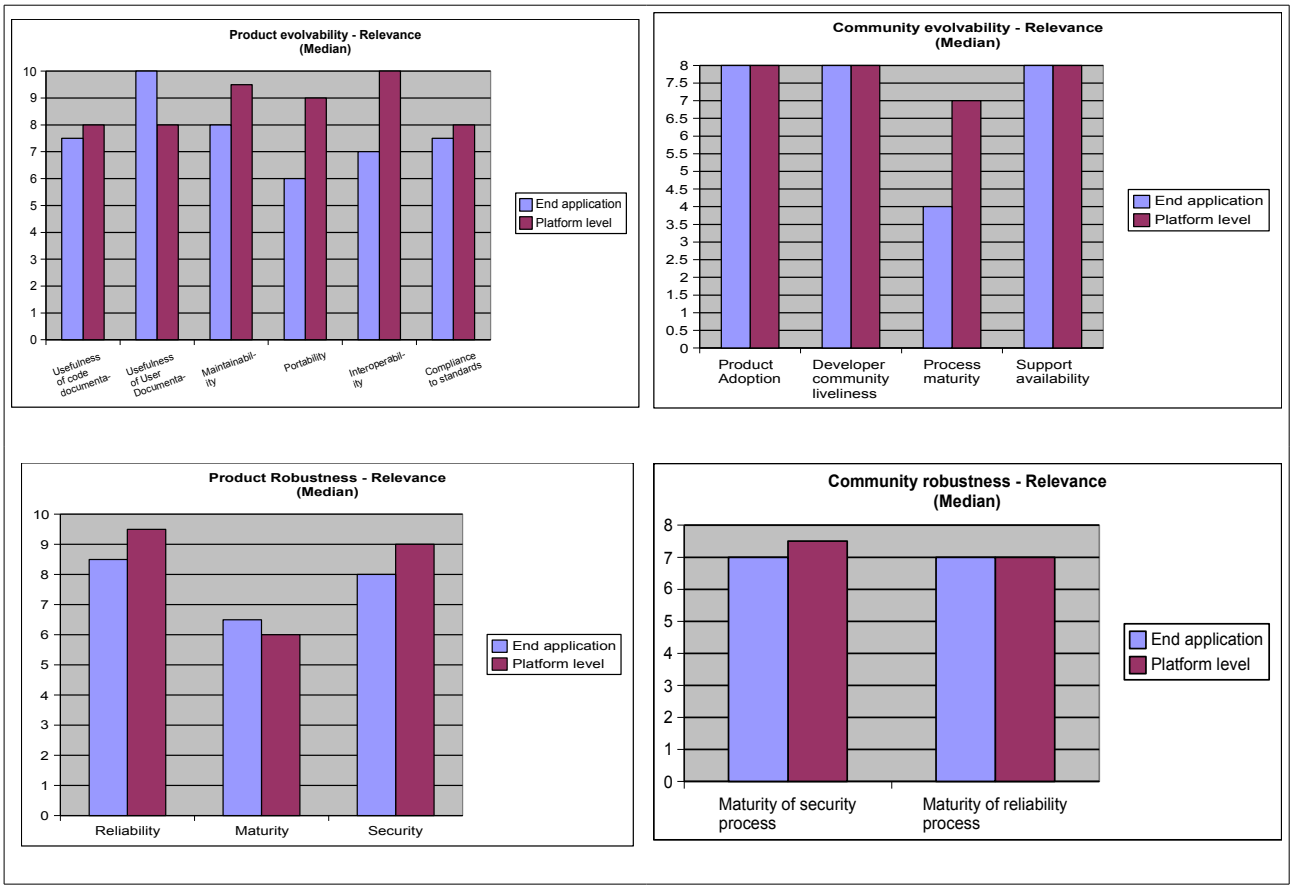


Figure 4: Relevance of quality characteristics.

3.3 Scenarios for case studies

At the end of the interview, the interviewer asked the F/OSS evaluators to describe their potential uses in terms of specific scenarios for which they would like to have support when evaluating F/OSS components. The following scenarios were mentioned:

Scenario 1: Selection, adoption and maintenance of a Wiki systems for internal use in a company.

Scenario 2: Selection of an appropriate graphical toolkit library.

Selection 3: Evaluation of less known and risky products.

In general, the interviews show that respondents differ on the proposed scenarios. Nevertheless, their opinion converges on the need to get support in the evaluation of the community. The main reason is that they have greater difficulties to extract the information necessary for assessing this aspect.

3.4 Discussion

Because of the limited number of conducted interviews and the use of a convenience sample, the results described above are neither conclusive nor complete. Nevertheless they represent a starting point for improving the definition of the first release of the QualOSS model and for planning and scoping the case studies in Workpackage 5.

Regarding quality characteristics to refine, it is important to solve the detected ambiguities in the definition of the QualOSS model before conducting any case studies (See Section 3.1). On the other hand, the list of missing quality characteristics is not conclusive: Each respondent suggests different quality characteristics. Before including any additional quality characteristics, a larger period of use of the QualOSS model and a larger evaluation of its completeness are advisable

Regarding the relevance of the quality characteristics, the results do not show a significant difference between the quality characteristics or the granularity level of the F/OSS component. That is probably because a larger sample is necessary for identifying significant differences.

Finally, from the perspective of planning scenarios for future case studies, the results suggest that the scenarios should focus on the community assessment.

It is also important to mention that the final decision on the improvement of the QualOSS model and the selection of scenarios for case studies requires an additional discussion with F/OSS practitioners.

4 Evaluation of an Additional Project with the 1st Release.

One of the activities included in Task 1.6 was the manual application of the first release of QualOSS quality model to an additional F/OSS project. The objectives of this activity were twofold:

- To determine if the model can be applied to a project lying out of the scope of the projects analysed so far.
- To collect additional experience with the use of the model, and, in particular, make sure that no major problems arise when using the model on an arbitrary F/OSS project.

Deliverable 1.4 lists the requirements for our original evaluation of five F/OSS projects, as well as the criteria used to select t those projects. Those projects were the Plone content management framework, the GNAT Pro Ada Compiler, the Hildon Application Framework, the JavaCC compiler generator and the Swallow service platform.

For our validation issue, the present, additional evaluation selected a sixth project, namely, the popular Asterisk telephony system. Asterisk can be used as a replacement for a standard PBX, as well as a server for Interned-based, IP telephony, and it includes a number of advanced features for applications such as call centres or specialized media servers. Asterisk runs on a variety of Unix-like operating systems and supports a wide array of telephony and network hardware. Asterisk is mainly coded in the C programming language.

The reason to select Asterisk is that it displays a vector of characteristics that clearly differs from those of our five previously analysed projects. Precisely, the fact of being a large server application coded in C and having real-time requirements differentiates Asterisk from our previously analysed projects, thus expanding the coverage of our validation.

The detailed results of the analysis are available in the spreadsheet that accompanies this document. Deliverable 1.5 includes an extensive list of the problems encountered while applying our prototype quality model. The application of this model to Asterisk did not show any major additional problems outside of this list. This increases our confidence that the model can be used (with the current limitations as discussed in D1.5) on a wider variety of projects.

5 Evaluation of Data Mining Methodologies

The work presented in this section focuses on specifying recommendations in order to help structure and organize the received raw metric data files into convenient databases that are well suited to carry out later deeper analyses using statistical tools as well as more advanced data mining methods. This effort was motivated by two important facts:

1. The number of analysed F/OSS projects in the prototypical phase was really small. It includes four distinct projects, namely HAFMaemo, JavaCC, Plone and SwallowDBE).

2. The quantity of metrics data made available for these four projects was relatively small. A great part of the metrics values are still missing.

For these reasons, it was impossible to start applying advanced data mining methods during Task 1.6. Therefore the main responsibility within task 1.6 concerning the validation of data mining aspects consisted in specifying recommendations and advices relative to the following points:

- The consolidation (fusion) work needed to bring together all the metrics data,
- The databases structuring (one separate database being built for each quality goal),
- The problems that we will have to tackle in order to be able to keep the greatest possible number of usable data. For instance, how does one have to handle “Not Applicable” or missing information?

5.1 Metric Data Merging

Three distinct databases were structured, one for each relevant quality goal (that is, Product Robustness, Product Evolvability and Community Evolvability). For reminder, there was no real need to consolidate into one database containing metrics data regarding the Community Robustness, given that very few metrics values were obtained for that particular quality goal.

With this purpose, the metrics data contained in the received spreadsheet files were merged together according to (1) the name of the metric and (2) the name of the quality goal (for a given spreadsheet, there were several tabs, each of them being associated with one of the 4 quality goals).

Figure 5 depicts a sample of the result of this fusion work for the Product Robustness quality goal. The quality metrics labels are provided along the first line of this database. For reminder, the metrics are considered as distinct variables (the so-called "attributes" in our Data Mining jargon).

Object_ID	Project_ID	Release_ID	AvailIssuesAllReleases	AvailIssuesSubsetReleases	ConfidIssuesAllReleases	...
1	HAFMaemo	HAF_All	7	NA	0	
2	HAFMaemo	HAF_20-21	NA	1	NA	
3	JavaCC	JavaCC_All	3	NA	0	
4	JavaCC	JavaCC_32	NA	0	NA	
5	JavaCC	JavaCC_40	NA	0	NA	
6	Plone	Plone_All	77	NA	0	
7	Plone	Plone_253	NA	0	NA	
8	Plone	Plone_21-21x	NA	45	NA	
9	Swallow	Swallow_All	0	NA	0	

Figure 5: Sample of the Product Robustness database.

A specific attribute, namely *Object_ID*, was added to the database structure. This attribute serves as the primary key of the database in order to identify the records in a one-to-one manner.

In that way, the database collects together several objects (a whole row representing a given object).

Note that a specific attribute, namely Project_ID, was also added to this database structure in order to track the name of the F/OSS project from which the objects come from. This can be helpful in the course of the Data Mining analysis if one wishes to put the focus on a particular F/OSS project.

For the two quality goals relative to the product (that is, Product Robustness and Product Evolvability), some metrics data were provided not only for the F/OSS product merging all releases but also for particular sub-releases of that same F/OSS product.

To be able to track this information, a specific attribute, namely *Release_ID,* conveying the name of the release was again added to the database structure. This attribute enables filtering actions if one wishes to put the focus on a given project sub-release.

In short, one particular object of this database (a unique value taken by the attribute *Object_ID*) represents one sub-release instance (attribute *Release_ID*) for one F/OSS product (attribute *Project_ID*). The remaining cells for that corresponding row convey includes the metrics values strictly speaking.

Figure 6 shows a sample of the database obtained after the fusion work for the Community Evolvability quality goal.

Similarly to the two databases associated with the two quality goals regarding the product, the database built for the Community Evolvability quality goal comprises additional attributes in order to:

- identify each record in a one-to-one way (role of *Object_ID*),
- put the focus on a given F/OSS project (role of *Project_ID*),
- identify the particular quarter time period (role of *Quarter_ID*),
- specify the start and end limits of each time interval (role of *InitPeriod* and *EndPeriod,* whose values are expressed here in an universal time format).

Object_ID	Project_ID	Quarter_ID	InitPeriod	EndPeriod	ActiveDev	NonActiveDev	...
1	HAFMaemo	1	38353	38443	0	0	
2	HAFMaemo	2	38443	38534	1	0	
3	HAFMaemo	3	38534	38626	9	0	
4	HAFMaemo	4	38626	38718	13	1	
5	HAFMaemo	5	38718	38808	13	4	
6	HAFMaemo	6	38808	38899	20	2	
7	HAFMaemo	7	38899	38991	23	8	
8	HAFMaemo	8	38991	39083	23	9	
9	HAFMaemo	9	39083	39173	22	12	
10	HAFMaemo	10	39173	39264	26	9	
11	HAFMaemo	11	39264	39356	18	17	
12	JavaCC	1	37712	37803	4	0	
13	JavaCC	2	37803	37895	1	3	
14	JavaCC	3	37895	37987	1	3	
15	JavaCC	4	37987	38078	1	3	
16	JavaCC	5	38078	38169	1	3	
17	JavaCC	6	38169	38261	1	3	
18	JavaCC	7	38261	38353	1	3	
19	JavaCC	8	38353	38443	2	3	
20	JavaCC	9	38443	38534	2	3	
…	…	…	…	…	…	…	

Figure 6: Sample of the Community Evolvability database.

In this way, the time information made available within the original spreadsheet files is preserved. Indeed, this detailed information was unfolded by repeating the corresponding project objects several times.

To summarize, three distinct databases were built according to the specific recommendations made within deliverable D1.5:

- The first one is associated with metrics data relative to the Product Robustness quality goal, and is completed with three supplementary attributes that serve as primary keys (namely, *Object_ID*, *Project_ID* and *Release_ID*).

- The second one gathers together metrics data regarding the Product Evolvability quality goal, and is completed with the three same primary key attributes as for the Product Robustness database.
- The third one collects metrics data relative to the Community Evolvability quality goal, and is completed with five supplementary attributes that serve as primary keys (i.e. *Object_ID*, *Project_ID, Quarter_ID, InitPeriod* and *EndPeriod*).
- As planned, no database was built to gather metrics data regarding the Community Robustness quality goal (given that the number of such metrics values was really insufficient).

5.2 Encountered problems during merging

In the course of the metrics data fusion work, it was necessary to pay attention to some encoding flaws keeping in mind our final goal, namely *to analyse the merged data through advanced Data Mining techniques* in an unambiguous way.

Some of the problems encountered and the corresponding remedies we provided are briefly described here.

1. **Metrics data taking the shape of text strings**

Figure 7 shows an example of this concern. The value filled in for the metric *ToolSupport* (linked to the Community Evolvability quality goal) is neither a numerical value nor a symbolic value but rather a free text entry.

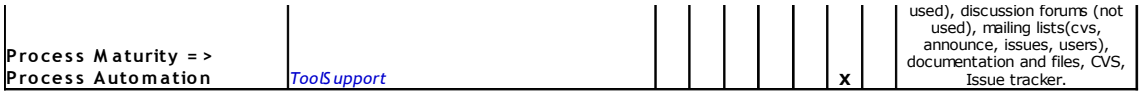


Figure 7: Example of metric value being a text string.

Given that, our Data Mining toolbox is not orientated towards Text Mining applications, in such cases, we recommend to use only well defined symbolic entries, that is, a restricted number of possible symbolic values (for instance, “forum”, “CVS” or “website” for the characteristic “process automation”.

2. **Metrics data taking the shape of a list of values**

Figure 8 illustrates an example of this problem. The value filled in for the metric *PeopleOnGroupsOfFiles* (linked again to the Community Evolvability quality goal) consists of a collection of numerical values.

Figure 8: Example of metric data being a collection of values.

In such situation, we recommend either to select a representative value (for instance, the mean) or to unwind the information conveyed by this numerical list using one of the appropriate primary key attributes (that is, reproduce several times the same object row within the database and then merely change the value taken by the attribute of interest).

3. **Metrics data mixing up symbolic and numerical values**

Figure 9 shows an example of this concern. The value filled in for the metric *APIDocumentationExistence* (connected with the Product Evolvability quality goal) can be either numerical (value *1*) or symbolic (value *yes*).

Figure 9: Example of metric data being either numerical or symbolic.

The two formats are of course correct (they express the same binary information, *0/1* or *no/yes*, in response to a question) but one must decide to use only one format at a time in order to avoid format confusion when building the databases.

5.3 First Graphic results

After the fusion work, the three built databases (still spreadsheets files) were loaded into our Data Mining toolbox (namely, the PEPITo® software). For each database, a PEPITo *project file* was created.

In fact, the *project file* is used to store and track the structure of the original spreadsheet database (how many attributes and objects form the database?, what is the data format of each of these attributes? etc.). Of course, this project file can be updated in order to take into account structural changes brought to the database (for instance, removing certain attributes or changing the data format of other ones) but also to store user-defined subsets of objects and/or attributes.

A sample of the PEPITo *project file* obtained for the Product Robustness database is shown in Figure 10.

```
<PROJECT NAME="Product-Robustness" CLASS="CORE.PROJECT::PROJECT" TEMPORAL-ATT="" >
<DRIVER NAME="buffer-for-driver" CLASS="DRIVER.DRIVER-BUFFER::DRIVER-BUFFER" >

<DRIVER NAME="driver" CLASS="DRIVER.DRIVER-EXCEL::DRIVER-EXCEL" >

<PROPERTY VALUE="row" NAME="headertype" />

<PROPERTY VALUE="Sheet1" NAME="sheetname" />

<PROPERTY VALUE="D:\Qualoss\Fichiers-Donnees\Product-Robustness-Database.xls" NAME="source" />

</DRIVER>

<PROPERTY VALUE="on-load" NAME="strategy" />

</DRIVER>

<ATTRIBUTE NAME="VulSubsetReleases" CLASS="DRIVER.ATTRIBUTE::ATTRIBUTE-DRIVER" >

<PROPERTY NAME="index" VALUE="50" />

<PROPERTY NAME="type" VALUE="NUMERICAL" />

<PROPERTY NAME="default" VALUE="0.0" />

<PROPERTY NAME="driver" VALUE="buffer-for-driver" />

<PROPERTY NAME="doc" VALUE="" />

</ATTRIBUTE>

<ATTRIBUTE NAME="VulAllReleases" CLASS="DRIVER.ATTRIBUTE::ATTRIBUTE-DRIVER" >

<PROPERTY NAME="index" VALUE="49" />

<PROPERTY NAME="type" VALUE="SYMBOLIC" />

<PROPERTY NAME="default" VALUE="miss" />

<PROPERTY NAME="driver" VALUE="buffer-for-driver" />

<PROPERTY NAME="doc" VALUE="" />

</ATTRIBUTE>

        ...
<ATTRIBUTE NAME="AvailIssuesAllReleases" CLASS="DRIVER.ATTRIBUTE::ATTRIBUTE-DRIVER" >

<PROPERTY NAME="index" VALUE="3" />

<PROPERTY NAME="type" VALUE="SYMBOLIC" />

<PROPERTY NAME="default" VALUE="miss" />

<PROPERTY NAME="driver" VALUE="buffer-for-driver" />

<PROPERTY NAME="doc" VALUE="" />
```

```
</ATTRIBUTE>
<ATTRIBUTE NAME="Release_ID" CLASS="DRIVER.ATTRIBUTE::ATTRIBUTE-DRIVER" >
<PROPERTY NAME="index" VALUE="2" />
<PROPERTY NAME="type" VALUE="SYMBOLIC" />
<PROPERTY NAME="default" VALUE="miss" />
<PROPERTY NAME="driver" VALUE="buffer-for-driver" />
<PROPERTY NAME="doc" VALUE="" />
</ATTRIBUTE>
<ATTRIBUTE NAME="Project_ID" CLASS="DRIVER.ATTRIBUTE::ATTRIBUTE-DRIVER" >
<PROPERTY NAME="index" VALUE="1" />
<PROPERTY NAME="type" VALUE="SYMBOLIC" />
<PROPERTY NAME="default" VALUE="miss" />
<PROPERTY NAME="driver" VALUE="buffer-for-driver" />
<PROPERTY NAME="doc" VALUE="" />
</ATTRIBUTE>
<ATTRIBUTE NAME="Object_ID" CLASS="DRIVER.ATTRIBUTE::ATTRIBUTE-DRIVER" >
<PROPERTY NAME="index" VALUE="0" />
<PROPERTY NAME="type" VALUE="NUMERICAL" />
<PROPERTY NAME="default" VALUE="0.0" />
<PROPERTY NAME="driver" VALUE="buffer-for-driver" />
<PROPERTY NAME="doc" VALUE="" />
</ATTRIBUTE>
<ITEMSET NAME="test-set" CLASS="CORE.ITEM-SET.SIMPLE::ITEM-SET-SIMPLE" >
<PROPERTY NAME="type" VALUE="objects" />
<RULE NAME="all" CLASS="CORE.ITEM-SET.RULES::ITEM-SET-RULE" >
<PROPERTY NAME="ACTION" VALUE="all" />
</RULE>
</ITEMSET>
<ITEMSET NAME="learning-set" CLASS="CORE.ITEM-SET.SIMPLE::ITEM-SET-SIMPLE" >
<PROPERTY NAME="type" VALUE="objects" />
<RULE NAME="all" CLASS="CORE.ITEM-SET.RULES::ITEM-SET-RULE" >
<PROPERTY NAME="ACTION" VALUE="all" />
</RULE>
</ITEMSET>
<ITEMSET NAME="All Objects" CLASS="CORE.ITEM-SET.FULL::ITEM-SET-FULL" >
<PROPERTY NAME="read-only" VALUE="true" />
<PROPERTY NAME="type" VALUE="objects" />
```

```
<PROPERTY NAME="name" VALUE="All Objects" />

</ITEMSET>

<ITEMSET NAME="candidate-attributes" CLASS="CORE.ITEM-SET.SIMPLE::ITEM-SET-SIMPLE" >

<PROPERTY NAME="type" VALUE="attributes" />

<RULE NAME="all" CLASS="CORE.ITEM-SET.RULES::ITEM-SET-RULE" >

<PROPERTY NAME="ACTION" VALUE="all" />

</RULE>

</ITEMSET>

<ITEMSET NAME="All Attributes" CLASS="CORE.ITEM-SET.FULL::ITEM-SET-FULL" >

<PROPERTY NAME="read-only" VALUE="true" />

<PROPERTY NAME="type" VALUE="attributes" />

<PROPERTY NAME="name" VALUE="All Attributes" />

</ITEMSET>

<COLORMAP NAME="colormap" CLASS="UTILITIES.COLORMAP::COLORMAP" >

<PROPERTY NAME="default" VALUE="NIL" />

<PROPERTY NAME="densitybrightness" VALUE="1.0" />

<PROPERTY NAME="densitysaturation" VALUE="1.0" />

<PROPERTY NAME="densityhue" VALUE="255" />

<PROPERTY NAME="brightness" VALUE="1.0" />

<PROPERTY NAME="saturation" VALUE="1.0" />

<PROPERTY NAME="hue" VALUE="255" />

</COLORMAP>

</PROJECT>
```

Figure 10: Project file sample for the Product Robustness database.

In short, in such file each metric attribute is characterized by its name, its index (i.e. the column position of the metric in the spreadsheet database), its format type (*symbolic* or *numerical*), its default value (typically, *miss* for a symbolic attribute and *0.0* for a numerical one) and a possible documentation.

Of course, all these features can be modified/completed by the user within the PEPITo® environment during the Data Mining analysis sessions.

The following graphics allow one to realize that other possible concerns (than those already discussed above) may hobble the carrying out of the Data Mining analysis.

The histogram constructed for the *FirstSourceAge* metric (Product Robustness database) is depicted in Figure 11. The data format automatically chosen by PEPITo® for that metric is the symbolic one since the data vector (within the spreadsheet database) gathers together:

- three cells with numerical values (*1*, 2 and *4*),
- two empty cells (thus, replaced by the default symbol *miss*),
- three cells with not applicable scenarios (represented by the *NA* symbol),
- one cell comprising the *nok* symbol.

On the one hand, it is obvious that the *nok* symbol is an encoding error that must be corrected (and avoided in the future). On the other hand, the missing values (*miss* symbol) and the not applicable scenarios cannot be corrected, so it is necessary to be able to handle them as they are.

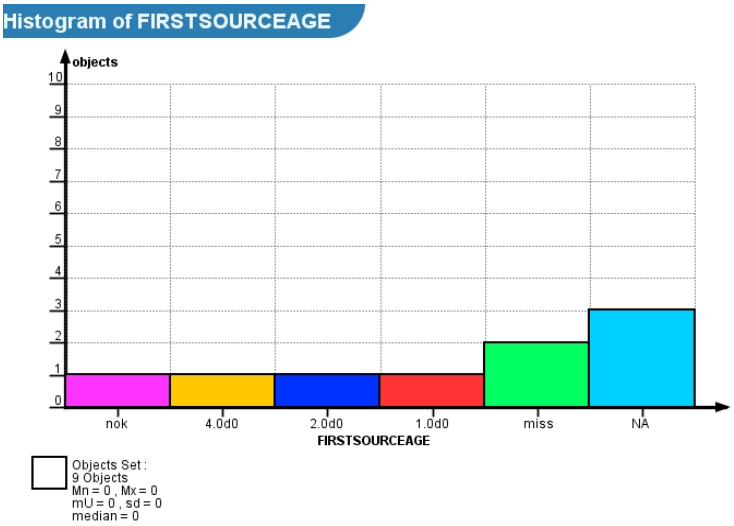


Figure 11: Example of metric value being a text string.Histogram of the FirstSourceAge metric.

Through PEPITo® environment, it is always possible to filter data and in particular missing (and/or not applicable) metrics values. Figure 12 illustrates the histogram obtained for the *AvailIssuesAllReleases* metric (Product Robustness database) once the *miss* and *NA* symbols are filtered.

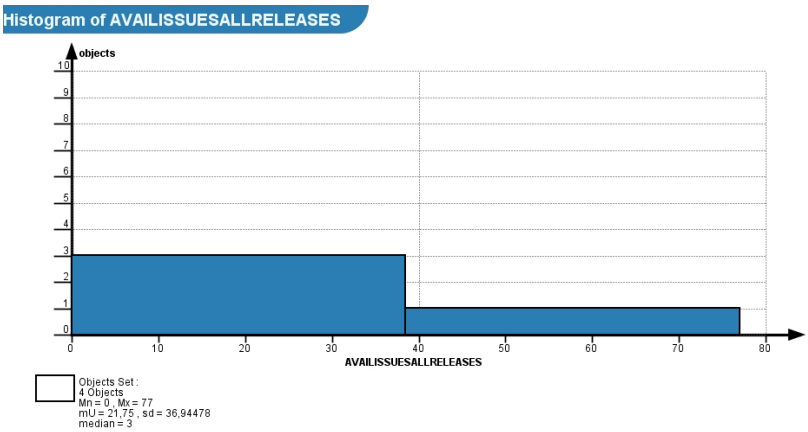


Figure 12: Histogram of the AvailIssuesAllReleases metric after filtering.

As indicated into the legend, the histogram was built using 4 different objects (5 objects are neglected through filtering). Since this attribute is of the numerical type, statistics are also provided (where *Mn*, *Mx*, *mU* and *sd* stand for minimum, maximum, mean and standard deviation respectively).

The histogram built for the *ActiveDev* metric (Community Evolvability database) is illustrated in Figure 13. For reminder, a database comprising 63 distinct objects was constructed from the 4 available F/OSS projects by unfolding the time information.

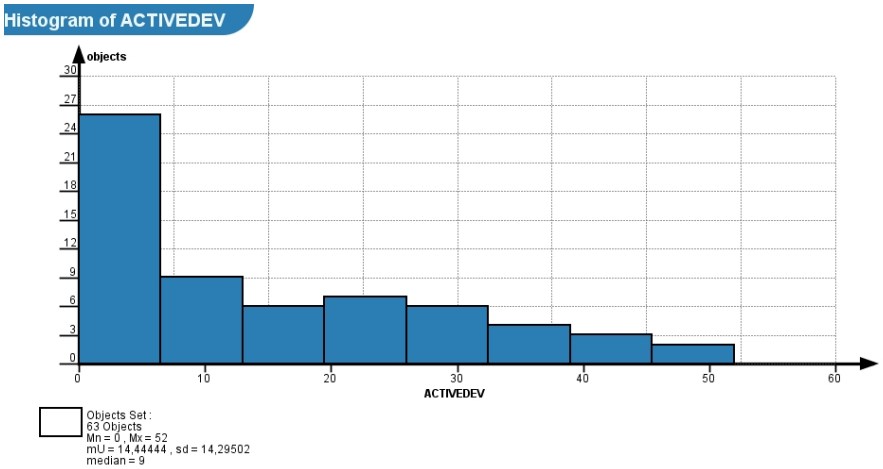


Figure 13: Histogram of the *ActiveDev* metric.

It is interesting to consider the impact of the F/OSS project information (conveyed by the *Project_ID* attribute) on this distribution. In this way, the Figure 14 illustrates a conditional histogram that gives an idea of the distribution of active developers with respect to the F/OSS product.

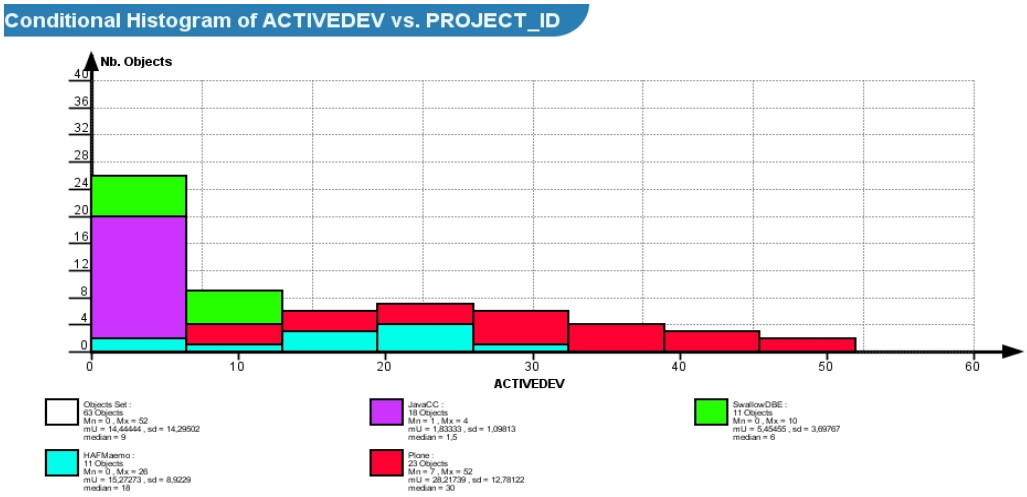


Figure 14: Histogram of the *ActiveDev* metric versus *Project_ID* attribute.

The distribution giving the number of different objects connected with each F/OSS project is illustrated in the Figure 15.

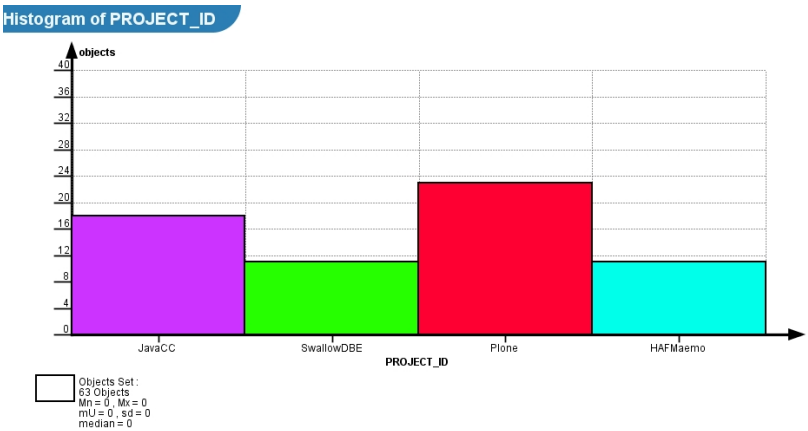


Figure 15: Histogram of the Project_ID attribute.

The scatter-plot drawn using the two metrics *MethodCyclomaticComplexityAverage* and *MethodLinesOfCodeAverage* (Product Evolvability database) is depicted in Figure 16.

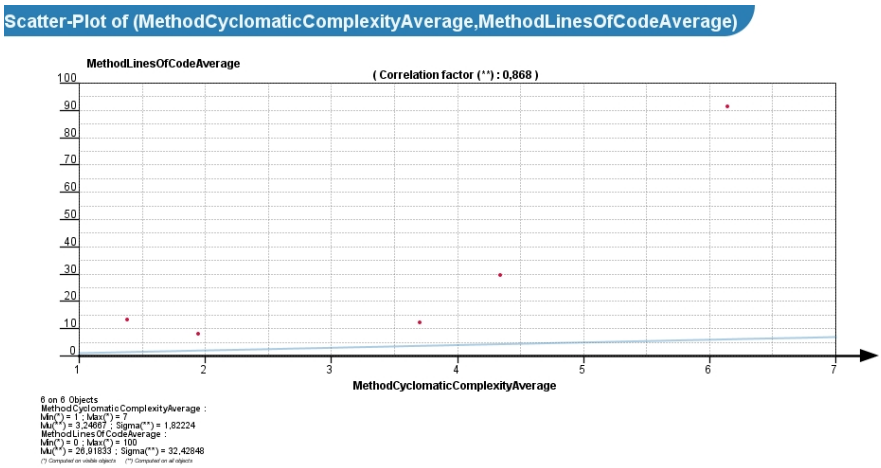


Figure 16: Example of scatter-plot drawn for the Product Evolvability database.

The correlation between these two metrics seems quite good (the linear correlation coefficient value is equal to 0,868). Of course, this result must only be seen as a possible trend in view of the restricted number of separate objects we have in the current database. A solid conclusion could only be drawn using a much larger set of distinct F/OSS products.

5.4 Main conclusions

Here are some important remarks or conclusions with regard to the validation work that has be done during the task 1.6 concerning data mining aspects

- In spite of certain encountered encoding flaws, the recommended approach to merge the raw metrics data into well structured databases was applied and validated.
- It is clear that a rigorous Data Mining approach, strictly speaking, will only be considered when a much larger number of metrics data (that is to say more distinct F/OSS projects) will be made available.
- One important goal of the QualOSS project is to consider a larger collection of quality metrics in comparison to what is done in other analogous research projects in the same field (for instance,

FLOSS Metrics). Unfortunately, at the present time, the QualOSS project suffers from a lack of analysed F/OSS projects and thus a lack of metrics data to mine. Consequently, artificial intelligence techniques cannot yet extract reliable and robust results nor uncover new significant knowledge out of the current metrics data.

- For that same reason, some activities discussed in the deliverable D1.5 which were suggested for the following tasks of the QualOSS project were not achieved so far (in particular, the activity aiming to automatically construct a given number of clusters in order to possibly characterize F/OSS projects families).

- In conclusion, it could be useful (even essential from a Data Mining point of view) to merge in the short term metrics data from both QualOSS (large set of metrics, few F/OSS projects) and FLOSS Metrics (restricted set of simple metrics, large number of F/OSS projects) investigations. Obviously, this would necessitate some extra work to harmonize the different formats.

6 Summary and Conclusions

This report chooses and designs an evaluation approach for the appropriateness of the definition of the QualOSS model focusing on the understandability, completeness and relevance of the proposed quality characteristics for evolvability and robustness. In this document, based on the perception of F/OSS evaluators, we identified candidates quality characteristics to refine and scenarios to be considered during the planning of case studies (Workpackage 5).

Additionally, we have presented the evaluation of a new project using the first release of QualOSS model as well as a set of recommendations to help structure and organize raw metric data files into convenient databases suitable for statistical and data mining methods.

Regarding the evaluation of the QualOSS model, further work is still needed. Because of the limited number of conducted interviews, the final results are neither complete nor conclusive. Therefore, further evaluations are needed to address the completeness and relevance of the QualOSS model. The subsequent evaluation should also address (1) the usefulness and usability of the QualOSS model and (2) the validity and reliability (accuracy) of the model. These tasks will be covered in Workpackage 5, which is responsible for planning and conducting case studies.

The QualOSS model has also been used for evaluating an additional project in a context different from those of the projects considered in Deliverable 1.5. The experience was successful and shows that the QualOSS model is applicable to other domains.

Regarding the application of a data mining approach, this will only be realizable when a much larger number of metrics data (F/OSS projects) is available. One important goal of the QualOSS project is to consider a larger collection of quality metrics in comparison to what is done in analogous research projects in the same field (for instance, FLOSSMetrics). Unfortunately, the QualOSS project currently suffers from a lack of analysed F/OSS projects, thus a lack of metrics data to mine. Consequently, artificial intelligence techniques can neither extract reliable and robust results nor uncover new significant knowledge out of the current metrics data. One forseen solution approach (to be discussed) is to merge in the short term metrics data from both QualOSS (large set of metrics, few F/OSS projects) and FLOSSMetrics (restricted set of simple metrics, large number of F/OSS projects).

7 REFERENCES

[1] Wohlin, C. , Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A. "*Experimentation in Software Engineering: An Introduction*". Kluwer Academic Publishers, 2000.

[2] Floyd J. Fowler, Jr. *"Improving survey questions: Design and Evaluation".* Applied Social Research Methods Series, SAGE Publications, 1995.

[3] Fleiss, J. "*Statistical methods for rates and proportions*". Wiley Series in Probability and Mathematical Statistics, Second Edition, 1981.

APPENDIX A: SURVEY – EXTENDED VERSION

7.1 INTRODUCTION

The goal of the interview is to evaluate the definition of the QualOSS model (i.e. of the quality characteristics definition and prioritization) compared to the perception and intuition of QualOSS evaluators. This evaluation is correspond to tasks 1.5 and 1.6 in the QualOSS project

Explain to interviewee: the main objective and summary of the QualOSS project. The main purpose is how intuitive and relevant are the definitions of the characteristics and sub-characteristics related to evolvability and robustness. Therefore this questionnaire asks for the perception of the evaluator of different definitions and items. The quality model must be presented according the levels covered in the questionnaire.

QualOSS (Quality of Open Source Software) is a project which purpose is to build a methodology and a tool to asses the evolvability and robustness of F/OSS components. Therefore a definition of evolvability and robustness in terms of quality characteristic and sub-characteristics has been proposed.

The team of QualOSS is interesting in your opinion about the proposed definitions of evolvability/robustness. The results will be confidential and will be used to validate and to improve the current definitions.

7.2 ORGANIZATIONAL INFORMATION

Date	
Start time	
Company	
Context / Domain	
Interviewee Name	
Interviewee Position	
Interviewee contact (Email, Telephone ,...)	

1. Do you use F/OSS components as end-application: . Yes. No.

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

   . . . . .

   Embedded External service Internal service Desktop Development

2. Do you use F/OSS components as platform level (to build an application): . Yes. No.

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

   . . . . .

   Embedded External service Internal service Desktop Development

3.

4. How do you use F/OSS components

   - Integrating an F/OSS product into a company's infrastructure
   - Integrating an OSS product/components into a software product/system developed by a company
   - Forking an existing open source component
   - Extending an open source product to communicate with my product
   - Selecting an open source language and libraries to develop my product on

7.3 Introduction to the model

First explain the definition of evolvability and its decomposition into product and community evolvability. Then present the hierarchy of the model for the product evolvability. Give a printed copy of the model to the interviewee and introduce the questions.

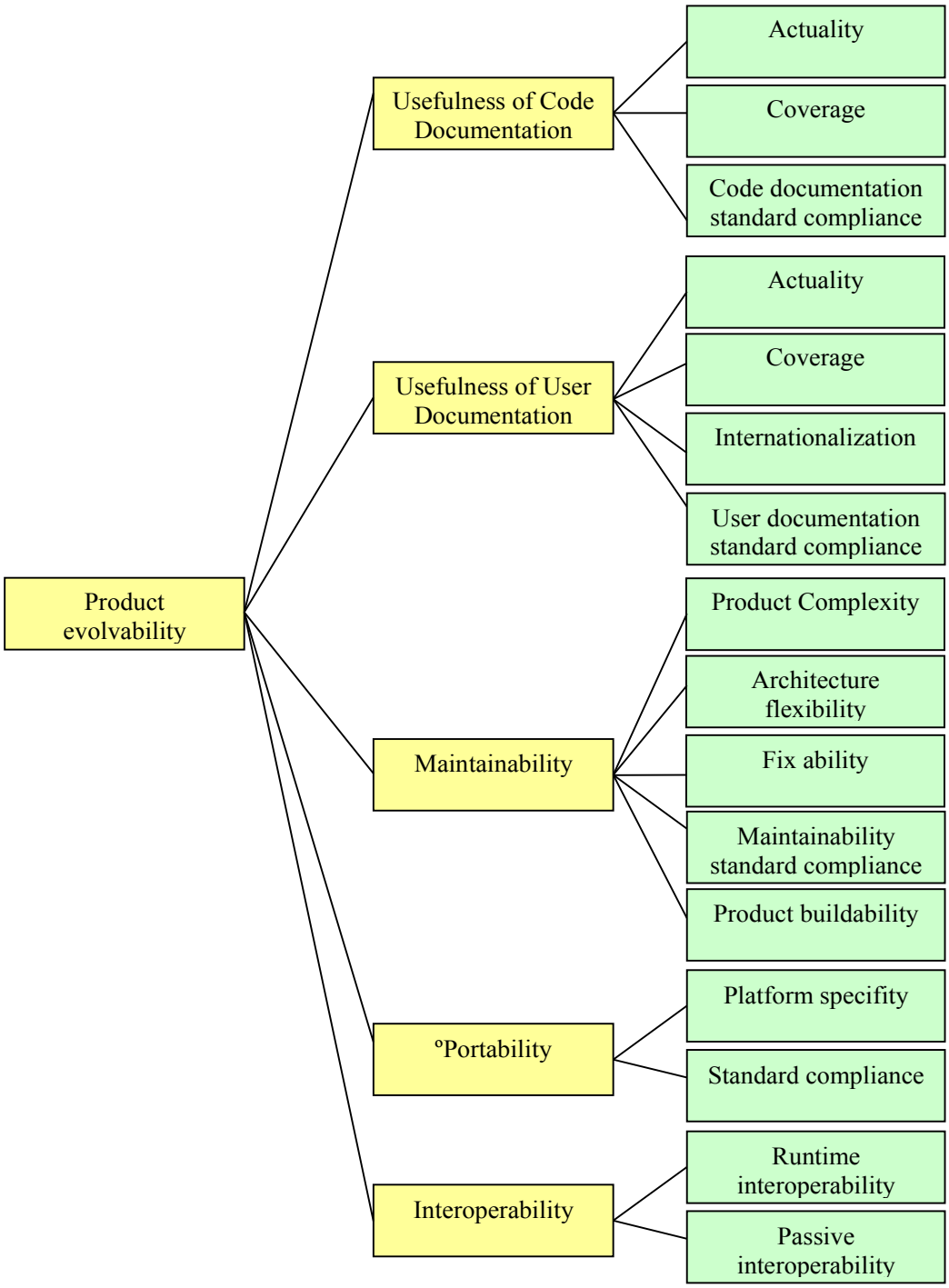
We define **evolvability** as the general ability of a F/OSS project to deliver useful products (or product updates) over an extended period of time. Also the ability of such products to remain useful for an extended period of time. In order to be able to decompose this wide notion into smaller criteria that can be studied separately, we consider products and their related F/OSS community independently from each other.

Now we will ask you to rank how meaningful and relevance the following characteristics are to assess the product evolvability. We will cover these characteristics according to the model hierarchy.

In order to evaluate meaningful, consider the scale completely meaningful, mostly meaningful, mostly meaningless and completely meaning less.

In order to evaluate relevance, consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant,

Please, let's assume you will evaluate OSS components, think about the criteria do you use and answer the following questions.


Product evolvability
Usefulness of Code Documentation
Actuality
Coverage
Code documentation standard compliance
Usefulness of User Documentation
Actuality
Coverage
Internationalization
User documentation standard compliance
Maintainability
Product Complexity
Architecture flexibility
Fix ability
Maintainability standard compliance
Product buildability
ºPortability
Platform specifity
Standard compliance
Interoperability
Runtime interoperability
Passive interoperability

7.4 Characteristic: Product Evolvability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Product evolvability: "The ability of a product to be corrected, adapted and extended over time, according to the needs of its users".	□	□	□	□	
1. Usefulness of code documentation: «Extent to which the source code documentation (explicitly describing the product's internals) is useful when performing corrections, adaptations or extensions to the product».	□	□	□	□	
2. Usefulness of user documentation: «Extent to which the product's user/administrator oriented documentation is useful when deploying and using the product».	□	□	□	□	
3. Maintainability: «Amount of effort required by a programmer or team of programmers with no previous knowledge of the product, to understand its code to the point that successful modifications are possible».	□	□	□	□	
4. Portability: «Ease with which a system or component can be transferred from one hardware or software environment to another».	□	□	□	□	
5. Interoperability: «Degree to which a software product can interoperate with other software product either live or based on input/output data ».	□	□	□	□	
6. Compliance to standards: «Degree to which a product complies with published standards that are relevant to its functionality. Important note: for measurement purposes, this criterion is applied separately to various relevant software artefacts, i.e., source code, documentation, etc»	□	□	□	□	

Question 2: When evaluating the product evolvability of OSS components, would you consider any additional characteristic?

No.	Characteristic	Description

Question 3: When evaluating OSS for use as end application, how relevant are these characteristics for you? *(For each additional characteristic, ask HOW RELEVANT IT IS?)*

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Usefulness of code documentation	10	9	8	7	6	5	4	3	2	1	0	
3. Maintainability	10	9	8	7	6	5	4	3	2	1	0	
4. Portability	10	9	8	7	6	5	4	3	2	1	0	
5. Interoperability	10	9	8	7	6	5	4	3	2	1	0	
6. Compliance to standards	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: When evaluating OSS for platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Usefulness of code documentation	10	9	8	7	6	5	4	3	2	1	0	
3. Maintainability	10	9	8	7	6	5	4	3	2	1	0	
4. Portability	10	9	8	7	6	5	4	3	2	1	0	
5. Interoperability	10	9	8	7	6	5	4	3	2	1	0	
6. Compliance to standards	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Characteristic: Usefulness of code documentation

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Actuality: «Extent to which the code documentation describes the current version of the source code as opposite to describing older versions of it»**.**	□	□	□	□	
2. Coverage: «Ratio between size of documented code and general product code size»	□	□	□	□	
3. Code documentation standard compliance: «Degree to which a product complies with published standards that are relevant to its functionality »	□	□	□	□	

Question 2: When evaluating the usefulness of code documentation of OSS components, would you consider any additional characteristics?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*				.....					*Completely irrelevant*		*Don´t Know*
1. Actuality	10	9	8	7	6	5	4	3	2	1	0	
2. Coverage	10	9	8	7	6	5	4	3	2	1	0	
3. Code documentation standard compliance	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: When evaluating OSS for platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Actuality	10	9	8	7	6	5	4	3	2	1	0	
2. Coverage	10	9	8	7	6	5	4	3	2	1	0	
3. Code documentation standard compliance	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

7.4.1 Characteristic: Usefulness of user documentation

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Actuality: «Extent to which the user documentation describes the current version of the product functionality as opposite to describing outdated functionality»..	□	□	□	□	
2. Coverage: «Ratio between the number of documented product features and the general number of features offered by the product»	□	□	□	□	
3. Internationalization: «Availability of the documentation in various natural languages»	□	□	□	□	
4. User documentation standard compliance: «Degree to which a product complies with published standards relevant to documentation»	□	□	□	□	

Question 2: When evaluating usefulness of user documentation of OSS components, would you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*				.....					*Completely irrelevant*		*Don´t Know*
1. Actuality	10	9	8	7	6	5	4	3	2	1	0	
2. Coverage	10	9	8	7	6	5	4	3	2	1	0	
3. Internationalization	10	9	8	7	6	5	4	3	2	1	0	
4. User documentation standard compliance	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: When evaluating OSS for platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Actuality	10	9	8	7	6	5	4	3	2	1	0	
2. Coverage	10	9	8	7	6	5	4	3	2	1	0	
3. Internationalization	10	9	8	7	6	5	4	3	2	1	0	
4. User documentation standard compliance	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

7.4.2 Characteristic: Maintainability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Product complexity: «Degree which system or component has a design or implementation that is difficult to understand and verify»	□	□	□	□	
2. Architecture flexibility: «Ability of the product's architecture of being applied to new problems. The ease which with a system or component can be modified for use in applications or environments other than those for which it was specifically designed»	□	□	□	□	
3. Fix ability: «Ease with which a software product can be fixed»	□	□	□	□	
4. Maintainability standard compliance: «Degree to which a product complies with published standards relevant to maintainability»	□	□	□	□	
5. Product buildability: «Degree to which a system or component can be rebuild after modifications to the source»	□	□	□	□	

Question 2: When evaluating the maintainability of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*					.....				*Completely irrelevant*		*Don't Know*
1. Product complexity	10	9	8	7	6	5	4	3	2	1	0	
2. Architecture flexibility	10	9	8	7	6	5	4	3	2	1	0	
3. Fix ability	10	9	8	7	6	5	4	3	2	1	0	
4. Maintainability standard compliance	10	9	8	7	6	5	4	3	2	1	0	
5. Product buidlanility	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: When evaluating OSS for platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*					.....				*Completely irrelevant*		*Don't Know*
1. Product complexity	10	9	8	7	6	5	4	3	2	1	0	
2. Architecture flexibility	10	9	8	7	6	5	4	3	2	1	0	
3. Fix ability	10	9	8	7	6	5	4	3	2	1	0	
4. Maintainability standard compliance	10	9	8	7	6	5	4	3	2	1	0	
5. Product buidability	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

7.4.3 Characteristic: Portability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Platform specificity: «Degree to which a product's code is specific to a particular hardware or software environment»	□	□	□	□	
2. Standard compliance: «Degree to which a product complies with published standards that are relevant to its functionality»	□	□	□	□	

Question 2: When evaluating the portability of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristics for you? (For each additional characteristic, ask HOW RELEVANT IT IS?)

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Platform specificity	10	9	8	7	6	5	4	3	2	1	0	
2. Standard compliance	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: When evaluating OSS for use as platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Platform specificity	10	9	8	7	6	5	4	3	2	1	0	
2. Standard compliance	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Characteristic: Interoperability

Question 1: How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Runtime interoperability: «Interoperability with other software products while in operation»	□	□	□	□	
2. Passive interoperability: «Interoperability with other software products based on output data generated by the software product or based on the capacity of the software product to read various data types and formats»	□	□	□	□	

Question 2: When evaluating the interoperability of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Runtime interoperability	10	9	8	7	6	5	4	3	2	1	0	
2. Passive interoperability	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

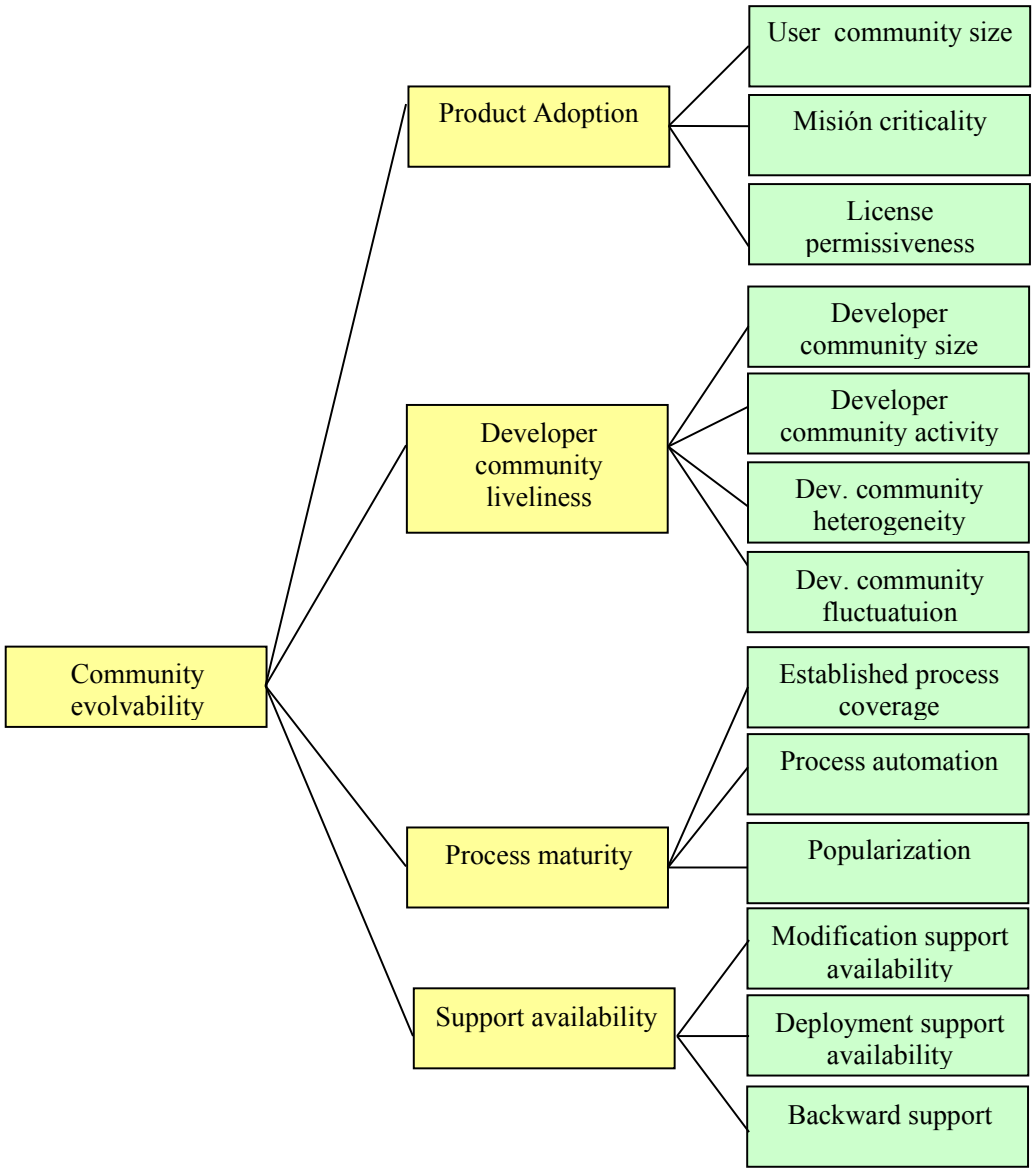
Question 4: When evaluating OSS for use as platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Runtime interoperability	10	9	8	7	6	5	4	3	2	1	0	
2. Passive interoperability	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

7.5 Characteristic: Community evolvability

Remind the responder that the definition of evolvability covers the the product and community evolvability. Then present the hierarchy of the model for the community evolvability. Give a printed copy of the model to the interviewee and introduce the questions.

We define evolvability as the general ability of a F/OSS to deliver useful products (or product updates) over an extended period of time. Also the ability of such products to remain useful for an extended period of time. In order to be able to decompose this wide notion into smaller criteria that can be studied separately, we consider products and their related F/OSS community independently from each other.



Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Community Evolvability: «The likelihood that a F/OSS community remains able to maintain the product or products it develops over an extended period of time».	□	□	□	□	
1. Product adoption: «Extent to which a F/OSS product is actively used by individuals and organization around the world».	□	□	□	□	
2. Developer community liveness: «Amount of work put by a development community into the creation and further development of a software product over a certain period of time».	□	□	□	□	
3. Process maturity: «Ability of a developer community to achieve development related goals by following established processes. Additionally, the level to which the processes followed by a development community are able to guarantee that certain desired product characteristics will be present in the product».	□	□	□	□	
4. Support availability «Ease with which a user can get support (e.g., engage experienced individuals or organizations) to perform tasks that make it possible to use a product for a particular purpose».	□	□	□	□	

Question 2: When evaluating the community evolvability OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Product adoption	10	9	8	7	6	5	4	3	2	1	0	
2. Developer community liveness	10	9	8	7	6	5	4	3	2	1	0	
3. Process maturity	10	9	8	7	6	5	4	3	2	1	0	
4. Support availability	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: When evaluating OSS for use as platform level, how relevant are these characteristics for you? ***For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Product adoption	10	9	8	7	6	5	4	3	2	1	0	
2. Developer community liveness	10	9	8	7	6	5	4	3	2	1	0	
3. Process maturity	10	9	8	7	6	5	4	3	2	1	0	
4. Support availability	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

7.5.1 Characteristic: Product adoption

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. User community size: «Number of users (individuals or organizations) that use a F/OSS product worldwide»**.**	□	□	□	□	
2. Mission criticality: «Extent to which users of a product apply it to mission-critical tasks. Alternatively, the degree to which users of a product depend on the product for reaching their business goals»	□	□	□	□	
3. License permissiveness: «Amount of freedom allowed to product users by the product's licence»	□	□	□	□	

Question 2: When evaluating the product adoption of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*	.....								*Completely irrelevant*		*Don´t Know*
1. User community size	10	9	8	7	6	5	4	3	2	1	0	
2. Mission criticality	10	9	8	7	6	5	4	3	2	1	0	
3. License permissiveness	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*				.....					*Completely irrelevant*		*Don´t Know*
1. User community size	10	9	8	7	6	5	4	3	2	1	0	
2. Mission criticality	10	9	8	7	6	5	4	3	2	1	0	
3. License permissiveness	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

7.5.2 Characteristic: Developer community liveness

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Developer community size: «Number of individuals and organizations actively contributing to a product's development over a certain period of time»**.**	□	□	□	□	
2. Developer community activity: «General number and size of the contributions made to a product's development over a certain period of time»	□	□	□	□	
3. Developer community heterogeneity: «Degree to which different types of developers (e.g., individuals vs. organizations, for-profit vs. non-for-profit, hobbyist vs paid professionals) are present in a developer community»	□	□	□	□	
4. Developer community fluctuation: «Rate movement of people into, and out of a developer community over time»	□	□	□	□	

Question 2: When evaluating the development community liveness of OSS components, do you consider any additional characteristic ?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Developer community size.	10	9	8	7	6	5	4	3	2	1	0	
2. Developer community activity	10	9	8	7	6	5	4	3	2	1	0	
3. Developer community heterogeneity	10	9	8	7	6	5	4	3	2	1	0	
4. Developer community fluctuation	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Developer community size.	10	9	8	7	6	5	4	3	2	1	0	
2. Developer community activity	10	9	8	7	6	5	4	3	2	1	0	
3. Developer community heterogeneity	10	9	8	7	6	5	4	3	2	1	0	
4. Developer community fluctuation	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

7.5.3 Characteristic: Process maturity

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Established process coverage: «Degree to which the development activities a community performs are covered by established, repeatable processes that are widely known and accepted by community members. Development processes that have been observed to be well established in existing development communities include project management, quality assurance and requirement engineering»**.**	□	□	□	□	
2. Process automation: «Degree to which established processes are partially or completely automated though the use of software tools. Examples of software tools commonly used by development communities to automate software processes include bug tracking systems, build farms and build daemons, and automated test suites»	□	□	□	□	
3. Popularization: « Availability of support related to popularize a software product ».	□	□	□	□	

Question 2: When evaluating the process maturity of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*	.....								*Completely irrelevant*	*Don´t Know*
1. Established process coverage	10	9	8	7	6	5	4	3	2	1	0	
2. Process automation	10	9	8	7	6	5	4	3	2	1	0	
3. Popularization	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Established process coverage	10	9	8	7	6	5	4	3	2	1	0	
2. Process automation	10	9	8	7	6	5	4	3	2	1	0	
3. Popularization	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

7.5.4 Characteristic: Support availability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Modification support availability: «Availability of support related to performing specific modifications to a software product»**.**	□	□	□	□	
2. Deployment support: «Availability of support related to solving problems arising from the deployment and use of a software product»	□	□	□	□	
3. Backward support: «Availability of support related to older version of a software product still in use»	□	□	□	□	

Question 2: When evaluating the support availability of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*relevant*					*Completely Completely ..... irrelevant*						*Don´t Know*
1. Modification support availability	10	9	8	7	6	5	4	3	2	1	0	
2. Deployment support	10	9	8	7	6	5	4	3	2	1	0	
3. Backward support	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

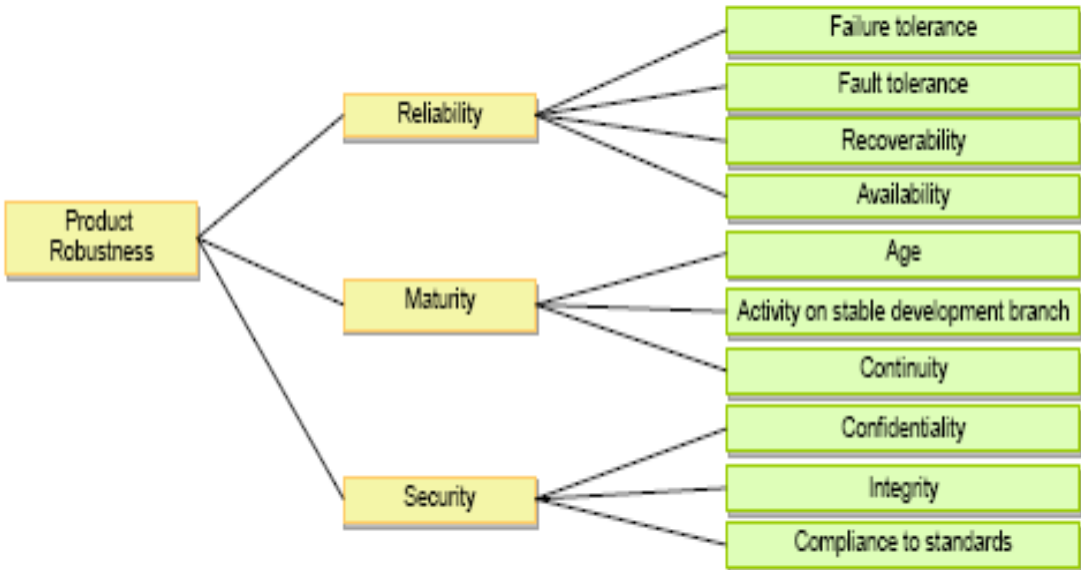
Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *irrelevant*											*Don´t Know*
1. Modification support availability	10	9	8	7	6	5	4	3	2	1	0	
2. Deployment support	10	9	8	7	6	5	4	3	2	1	0	
3. Backward support	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

To close the interview, ask the responder to any additional COMMENTS:

8 Introduction to the model

First explain the definition of robustness and its decomposition into product and community robustness. Then present the hierarchy of the model for the product robustness. Give a printed copy of the model to the interviewee and introduce the questions.



Now we will ask you to rank how meaningful and relevance the following characteristics are to assess the product robustness. We will cover these characteristics according to the model hierarchy.

In order to evaluate meaningful, consider the scale completely meaningful, mostly meaningful, mostly meaningless and completely meaning less.

In order to evaluate relevance, consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant,

Please, let's assume you will evaluate OSS components, think about the criteria do you use and answer the following questions.

8.1 Characteristic: Product Robustness

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Product Robustness: «Degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions».	□	□	□	□	
1. Reliability: «Ability of a system or component to perform its required functions under stated conditions for a specified period of time».	□	□	□	□	
2. Security: « Capability of the software product to protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them».	□	□	□	□	
3. Maturity: «Degree to which the general, long term objectives set for a product have been reached by current implementation».	□	□	□	□	

Question 2: When evaluating the product robustness of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Reliability	10	9	8	7	6	5	4	3	2	1	0	
2. Security	10	9	8	7	6	5	4	3	2	1	0	
3. Maturity	10	9	8	7	6	5	4	3	2	1	0	

	10	9	8	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *irrelevant*											*Don´t Know*
1. Reliability	10	9	8	7	6	5	4	3	2	1	0	
2. Security	10	9	8	7	6	5	4	3	2	1	0	
3. Maturity	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

8.1.1 Characteristic: Reliability

Explain the difference between failure and fault

1. ***A software failure occurs when the software doesn´t do what the user expects to see according the requirements.#***
2. ***A software fault occurs is an incorrect step, process, or data definition in a computer program.***

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Failure tolerance: «Capability of the software product to avoid failure as a result of faults in the software»**.**	□	□	□	□	
2. Fault tolerance: «Capability of the software product to maintain a specified level of performance in cases of software faults or of infringement of its specified interface»	□	□	□	□	
3. Recoverability: «Capability of the software product to re-establish a specified level of performance and recover the data directly affected in the case of a failure»	□	□	□	□	
4. Availability: «Degree to which a system or component is operational and accessible when required for use»	□	□	□	□	

Question 2: When evaluating the reliability of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Failure tolerance	10	9	8	7	6	5	4	3	2	1	0	
2. Fault tolerance	10	9	8	7	6	5	4	3	2	1	0	
3. Recoverability	10	9	8	7	6	5	4	3	2	1	0	
4. Availability	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Failure tolerance	10	9	8	7	6	5	4	3	2	1	0	
2. Fault tolerance	10	9	8	7	6	5	4	3	2	1	0	
3. Recoverability	10	9	8	7	6	5	4	3	2	1	0	
4. Availability	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

8.1.2 Characteristic: Security

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Confidentiality: «Degree to which a system prevents unauthorized disclosure of information; that is, provides assurance that information is not disclosed to unauthorized individuals, processes, or devices»**.**	□	□	□	□	
2. Integrity: «Degree to which a system or component is able to protect the accuracy and completeness of information and processing methods»	□	□	□	□	
3. Security standards compliance: «Degree to which a product complies with published security standards that are relevant to its functionality»	□	□	□	□	

Question 2: When evaluating the security of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*relevant* ..... *irrelevant*											*Don´t Know*
	Completely										*Completely*	
1. Confidentiality	10	9	8	7	6	5	4	3	2	1	0	
2. Integrity	10	9	8	7	6	5	4	3	2	1	0	
3. Security standards compliance	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Confidentiality	10	9	8	7	6	5	4	3	2	1	0	
2. Integrity	10	9	8	7	6	5	4	3	2	1	0	
3. Security standards compliance	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

8.1.3 Characteristic: Maturity

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Age: «Time span over which a product has been developed»**.**	□	□	□	□	
2. Continuity: «Regularity with which community contributions have been made to the a product or in relation to the product over its lifespan»	□	□	□	□	
3. Activity on stable development branch: «Number and size of the contributions made to a product's stable development branch over a certain period of time»	□	□	□	□	

Question 2: When evaluating the maturity of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

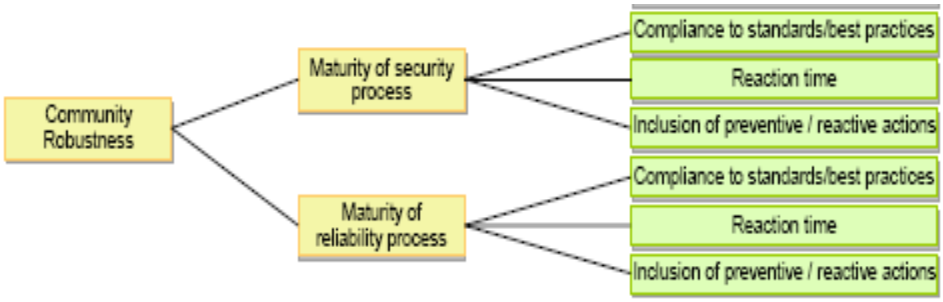
Sub-characteristic	*relevant*					*Completely Completely ..... irrelevant*						*Don´t Know*
1. Age	10	9	8	7	6	5	4	3	2	1	0	
2. Continuity	10	9	8	7	6	5	4	3	2	1	0	
3. Activity on stable development branch	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Age	10	9	8	7	6	5	4	3	2	1	0	
2. Continuity	10	9	8	7	6	5	4	3	2	1	0	
3. Activity on stable development branch	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

8.2 Characteristic: Community Robustness

Remenber the responder that the definition of robustness covers the the product and community robustness. Then present the hierarchy of the model for the community robustness. Give a printed copy of the model to the interviewee and introduce the questions.



Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Community robustness: «The ability of the established processes in a community to guarantee the delivery of robust products».	□	□	□	□	
1. Maturity of security process: «Degree to which a development community has established processes dedicated to guarantee the security of delivered products».	□	□	□	□	
2. Maturity of reliability process: «Degree to which a development community has established processes dedicated to guarantee that delivered products are free of critical defects (defects that prevent the operation of the product under common operation conditions)».	□	□	□	□	

Question 2: When evaluating the community robustness of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*										*Completely irrelevant*	*Don´t Know*
1. Maturity of security process	10	9	8	7	6	5	4	3	2	1	0	
2. Maturity of reliability process	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *irrelevant*											*Don´t Know*
1. Maturity of security process	10	9	8	7	6	5	4	3	2	1	0	
2. Maturity of reliability process	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

8.2.1 Characteristic: Maturity of the security process

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Compliance: «Degree to which the processes and procedures dealing with security adhere to best practices and security standards»**.**	□	□	□	□	
2. Reaction time: «Amount of time that is typically required for resolving security-related issues »	□	□	□	□	
3. Inclusion of preventive/reactive actions: «Degree to which the community commits to actions aimed at preventing security problems »	□	□	□	□	

Question 2: When evaluating the maturity of the security process of OSS components, do you consider any additional characteristic ?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*relevant*					*Completely* ..... *irrelevant*					*Completely*	*Don´t Know*
1. Compliance	10	9	8	7	6	5	4	3	2	1	0	
2. Reaction time	10	9	8	7	6	5	4	3	2	1	0	
3. Inclusion of preventive/reactive actions	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*										*Completely irrelevant*	*Don´t Know*
1. Compliance	10	9	8	7	6	5	4	3	2	1	0	
2. Reaction time	10	9	8	7	6	5	4	3	2	1	0	
3. Inclusion of preventive/reactive actions	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

8.2.2 Characteristic: Maturity of the reliability process

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Compliance: «Degree to which the processes and procedures dealing with security adhere to best practices and security standards»**.**	□	□	□	□	
2. Reaction time: «Amount of time that is typically required for resolving reliability-related issues »	□	□	□	□	
3. Inclusion of preventive/reactive actions: «Degree to which the community commits to actions aimed at preventing reliability problems »	□	□	□	□	

Question 2: When evaluating the maturity of the reliability processs of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*relevant*					*Completely* ..... *irrelevant*					*Completely*	*Don´t Know*
1. Compliance	10	9	8	7	6	5	4	3	2	1	0	
2. Reaction time	10	9	8	7	6	5	4	3	2	1	0	
3. Inclusion of preventive/reactive actions	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *irrelevant*											*Don´t Know*
1. Compliance	10	9	8	7	6	5	4	3	2	1	0	
2. Reaction time	10	9	8	7	6	5	4	3	2	1	0	
3. Inclusion of preventive/reactive actions	10	9	8	7	6	5	4	3	2	1	0	
	10	9	8	7	6	5	4	3	2	1	0	

To close the interview, ask the responder to any additional COMMENTS:

9 Business Goals

The goal of this part is to elicit potential usage scenarios of the QualOSS platform, and how we can focus our work to support users in future.

Question 1: In which situations or scenarios would you like to have the support of such an evaluation? Are there any specific aspects that you find difficult to evaluate now, and where you would like to have support from QualOSS?

<Hints for Interviewer: If the interviewee does not come up with ideas, give examples of what we want to know, e.g.:

Usage scenario class 1: Someone wants to use an open source component to build a system for a customer, and has to evaluate several potential alternatives. What kind of support would he like to get in evaluating OSS components? What are his largest problems in evaluating OSS components at the moment? (e.g., security assessment, supplier/community assessment, process maturity, can I get support for this project, ...)

Usage scenario class 2: someone wants to get support for implementing requirements in an OSS component. On a marketplace, several companies / developers have been contacted and are offering their services. What kind of support would you like to get in evaluating OSS service suppliers? (e.g., an assessment of quality and quantity of the competitors' contributions, ...)

For more information, see also "QualOSS_Goals.odt">

Question 2: Now, that we discussed business goals or usage scenarios, are there any additional characteristics that come to your mind?

APPENDIX B: INTERVIEW RESULTS

9.1 INTERVIEW 1

9.2 ORGANIZATIONAL INFORMATION

Date	January 2008
Start time	
Company	Organization A
Context / Domain	
Interviewee Position	

1. Do you use F/OSS components as end-application: X Yes. No.

   Uses the components developed by the company for development.

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

   . . . . .

   Embedded External service Internal service Desktop Development

2. Do you use F/OSS components as platform level (to build an application): X Yes. No

   Intw. uses floss tools for building internal systems in the company. Company integrates FlOSS into its infrastructure, but not role of the intw.

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

   . . . . .

   Embedded External service Internal service Desktop Development

3. How do you use F/OSS components

   . Integrating an F/OSS product into a company's infrastructure

   . Integrating an OSS product/components into a software product/system developed by a company

   . Forking an existing open source component

   . Extending an open source product to communicate with my product

   . Selecting an open source language and libraries to develop my product on

9.3 Characteristic: Product Evolvability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Product evolvability: "The ability of a product to be corrected, adapted and extended over time, according to the needs of its users".	X	☐	☐	☐	
1. Usefulness of code documentation: «Extent to which the source code documentation (explicitly describing the product's internals) is useful when performing corrections, adaptations or extensions to the product».	X	☐	☐	☐	Possible confusion between 1 and 2 here
2. Usefulness of user documentation: «Extent to which the product's user/administrator oriented documentation is useful when deploying and using the product».	X	☐	☐	☐	
3. Maintainability: «Amount of effort required by a programmer or team of programmers with no previous knowledge of the product, to understand its code to the point that successful modifications are possible».	X	☐	☐	☐	Is clear, but should include the effort necessary for actually changing the code. Do not restrict to understanding. Example, in assembly, understanding can be much easier than changing. The GNAT compiler is hard to understand, but changes may end up being trivial.
4. Portability: «Ease with which a system or component can be transferred from one hardware or software environment to another».	X	☐	☐	☐	Is good, all-inclussive.
5. Interoperability: «Degree to which a software product can interoperate with other software product either live or based on input/output data ».	☐	X	☐	☐	Self-referencing definition.
6. Compliance to standards: «Degree to which a product complies with published standards that are relevant to its functionality. Important note: for measurement purposes, this criterion is applied separately to various relevant software artefacts, i.e., source code, documentation, etc»	☐	☐	☐	☐	First question: what is a standard? It seems hard t understand which standards you're speaking about here. Depending on the standard, this may apply to other categories, I.e., quality of documentation.

Question 2: When evaluating the product evolvability of OSS components, would you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Usefulness of code documentation	10	9	8	7	6	5	X	3	2	1	0	
2. Usefulness of user documentation	X	9	8	7	6	5	4	3	2	1	0	
3. Maintainability	10	9	8	7	6	5	X	3	2	1	0	
4. Portability	10	9	8	7	6	X	4	3	2	1	0	
5. Interoperability	X	9	8	7	6	5	4	3	2	1	0	
6. Compliance to standards	10	9	8	X	6	5	4	3	2	1	0	

Question 4: When evaluating OSS for platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Usefulness of code documentation	10	9	X	7	6	5	4	3	2	1	0	
2. Usefulness of user documentation	10	9	X	7	6	5	4	3	2	1	0	
3. Maintainability	X	9	8	7	6	5	4	3	2	1	0	
4. Portability	X	9	8	7	6	5	4	3	2	1	0	
5. Interoperability	X	9	8	7	6	5	4	3	2	1	0	
6. Compliance to standards	10	9	8	X	6	5	4	3	2	1	0	

9.3.1 Characteristic: Usefulness of code documentation

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Actuality: «Extent to which the code documentation describes the current version of the source code as opposite to describing older versions of it»**.**	X	☐	☐	☐	
2. Coverage: «Ratio between size of documented code and general product code size»	☐	☐	☐	X	Not appropriate. Should be related to "all sections of code being described"
3. Code documentation standard compliance: «Degree to which a product complies with published standards that are relevant to its functionality »	☐	☐	X	☐	

Question 2: When evaluating the usefulness of code documentation of OSS components, would you consider any additonal characteristics?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Actuality	10	9	8	7	6	X	4	3	2	1	0	
2. Coverage	X	9	8	7	6	5	4	3	2	1	0	
3. Code documentation standard compliance	10	9	8	7	6	5	4	3	X	1	0	

Question 4: When evaluating OSS for platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Actuality	10	9	8	X	6	5	4	3	2	1	0	
2. Coverage	X	9	8	7	6	5	4	3	2	1	0	
3. Code documentation standard compliance	10	9	8	7	6	5	4	3	X	1	0	

The existence of automatic documentation generation makes it easier for it to be up to date.

9.3.2 Characteristic: Usefulness of user documentation

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	Completely meaningful	Mostly meaningful	Mostly meaningless	Completely meaningless	Comments
1. Actuality: «Extent to which the user documentation describes the current version of the product functionality as opposite to describing outdated functionality»..	□	□	□	□	
2. Coverage: «Ratio between the number of documented product features and the general number of features offered by the product»	□	□	□	□	
3. Internationalization: «Availability of the documentation in various natural languages»	X	□	□	□	
4. User documentation standard compliance: «Degree to which a product complies with published standards relevant to documentation»	□	□	□	□	

Question 2: When evaluating usefulness of user documentation of OSS components, would you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*			.....						*Completely irrelevant*		*Don´t Know*
1. Actuality	X	9	8	7	6	5	4	3	2	1	0	
2. Coverage	X	9	8	7	6	5	4	3	2	1	0	
3. Internationalization	10	9	8	7	6	5	4	3	2	1	X	
4. User documentation standard compliance	10	9	8	7	6	5	4	3	X	1	0	

Question 4: When evaluating OSS for platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*			.....						*Completely irrelevant*		*Don´t Know*
1. Actuality	X	9	8	7	6	5	4	3	2	1	0	
2. Coverage	X	9	8	7	6	5	4	3	2	1	0	
3. Internationalization	10	9	8	7	6	5	4	3	2	1	X	
4. User documentation standard compliance	10	9	8	7	6	5	4	3	X	1	0	

9.3.3 Characteristic: Maintainability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Product complexity: «Degree which system or component has a design or implementation that is difficult to understand and verify»	☐	☐	X	☐	Refinement is necessary, specially regarding the concept of complexity.
2. Architecture flexibility: «Ability of the product's architecture of being applied to new problems. The ease which with a system or component can be modified for use in applications or environments other than those for which it was specifically designed»	☐	X	☐	☐	Fussy, difficult to measure.
3. Fixability: «Ease with which a software product can be fixed»	X	☐	☐	☐	
4. Maintainability standard compliance: «Degree to which a product complies with published standards relevant to maintainability»	X	☐	☐	☐	Clarify which standards apply.
5. Product buildability: «Degree to which a system or component can be rebuild after modifications to the source»	X	☐	☐	☐	

Question 2: When evaluating the maintainability of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*
	Presence of regressive test suite	Important for maintainability: you can check changes.

Question 3: When evaluating OSS for use as end application, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*			.....						*Completely irrelevant*		*Don´t Know*
1. Product complexity	10	9	8	7	6	5	4	3	X	1	0	
2. Architecture flexibility	10	9	8	7	6	5	4	X	2	1	0	
3. Fix ability	10	9	8	7	X	5	4	3	2	1	0	
4. Maintainability standard compliance	10	9	8	7	6	5	X	3	2	1	0	
5. Product buidability	X	9	8	7	6	5	4	3	2	1	0	
Presence of regressive test suite	10	9	8	X	6	5	4	3	2	1	0	

Question 4: When evaluating OSS for platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*			.....						*Completely irrelevant*		*Don´t Know*
1. Product complexity	10	9	8	X	6	5	4	3	2	1	0	
2. Architecture flexibility	10	9	8	X	6	5	4	3	2	1	0	
3. Fix ability	X	9	8	7	6	5	4	3	2	1	0	
4. Maintainability standard compliance	10	9	X	7	6	5	4	3	2	1	0	
5. Product buidability	X	9	8	7	6	5	4	3	2	1	0	

Not much difference between understanding and making the change.

9.3.4 Characteristic: Portability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Platform specificity: «Degree to which a product's code is specific to a particular hardware or software environment»	X	□	□	□	
2. Standard compliance: «Degree to which a product complies with published standards that are relevant to its functionality»	X	□	□	□	Distinction between hardware, code and development environment compatibility.

Question 2: When evaluating the portability of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristics for you? (For each additional characteristic, ask HOW RELEVANT IT IS?)

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Platform specificity	10	9	8	7	6	5	4	3	X	1	0	
2. Standard compliance	10	9	8	7	6	5	4	3	X	1	0	

Question 4: When evaluating OSS for use as platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Platform specificity	10	9	8	X	6	5	4	3	2	1	0	
2. Standard compliance	10	9	8	X	6	5	4	3	2	1	0	

9.3.5 Characteristic: Interoperability

Question 1: How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Runtime interoperability: «Interoperability with other software products while in operation»	X	☐	☐	☐	Make it specific to the products used in a company. Absolute interop. is not necessary.
2. Passive interoperability: «Interoperability with other software products based on output data generated by the software product or based on the capacity of the software product to read various data types and formats»	X	☐	☐	☐	Same as above. You don't want to be locked to a particular product. Is harder to detect. Issues my take longer to be evident.

Question 2: When evaluating the interoperability of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Runtime interoperability	10	9	X	7	6	5	4	3	2	1	0	
2. Passive interoperability	X	9	8	7	6	5	4	3	2	1	0	

Question 4: When evaluating OSS for use as platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Runtime interoperability	10	9	X	7	6	5	4	3	2	1	0	
2. Passive interoperability	X	9	8	7	6	5	4	3	2	1	0	

9.4

9.5 Characteristic: Community evolvability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	Completely meaningful	Mostly meaningful	Mostly meaningless	Completely meaningless	Comments
Community Evolvability: «The likelihood that a F/OSS community remains able to maintain the product or products it develops over an extended period of time».	☐	X	☐	☐	Seems to correspond more to sustainability. Is still interesting. You may distinguish between maintaining a product running or really developing new functionality. Confusion may be related to the meaning of "maintain": related to small changes.
1. Product adoption: «Extent to which a F/OSS product is actively used by individuals and organization around the world».	X	☐	☐	☐	
2. Developer community liveness: «Amount of work put by a development community into the creation and further development of a software product over a certain period of time».	X	☐	☐	☐	
3. Process maturity: «Ability of a developer community to achieve development related goals by following established processes. Additionally, the level to which the processes followed by a development community are able to guarantee that certain desired product characteristics will be present in the product».	X	☐	☐	☐	Maturity sounds related to time, but the definition doesn't include it.
4. Support availability «Ease with which a user can get support (e.g., engage experienced individuals or organizations) to perform tasks that make it possible to use a product for a particular purpose».	X	☐	☐	☐	

Question 2: When evaluating the community evolvability OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*
	The level of involvement of a company is independent from its size. This may be a subcharacteristic of liveliness.	

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Product adoption	10	9	X	7	6	5	4	3	2	1	0	
2. Developer community liveness	10	9	8	7	6	X	4	3	2	1	0	
3. Process maturity	10	9	8	7	6	5	4	X	2	1	0	
4. Support availability	10	9	X	7	6	5	4	3	2	1	0	
Commitment of large companies	10	9	8	7	6	X	4	3	2	1	0	

Question 4: When evaluating OSS for use as platform level, how relevant are these characteristics for you? ***For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Product adoption	10	9	X	7	6	5	4	3	2	1	0	
2. Developer community liveness	10	9	8	X	6	5	4	3	2	1	0	
3. Process maturity	10	9	8	X	6	5	4	3	2	1	0	
4. Support availability	10	9	X	7	6	5	4	3	2	1	0	

9.6 Characteristic: Product Robustness

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	Completely meaningful	Mostly meaningful	Mostly meaningless	Completely meaningless	Comments
Product Robustness: «Degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions».	□	□	X	□	Robustness can also be an issue under ordinary conditions. 'Correctly' should mean 'behaves according to specification'. Robustness may be different depending on the platform.
1. Reliability: «Ability of a system or component to perform its required functions under stated conditions for a specified period of time».	X	□	□	□	

2. Security: « Capability of the software product to protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them».	X	□	□	□	
3. Maturity: «Degree to which the general, long term objectives set for a product have been reached by current implementation».	□	□	□	□	Alternative definition: "a product is mature when few further changes are needed". Probably remove "long term" from the definition. Maturity is not always a long-term issue. This may overlap with other characteristics.

Question 2: When evaluating the product robustness of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Reliability	X	9	8	7	6	5	4	3	2	1	0	
2. Security	X	9	8	7	6	5	4	3	2	1	0	
3. Maturity	10	9	8	7	6	X	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Reliability	X	9	8	7	6	5	4	3	2	1	0	
2. Security	X	9	8	7	6	5	4	3	2	1	0	
3. Maturity	10	9	8	7	6	X	4	3	2	1	0	

9.7 Characteristic: Community Robustness

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	Completely meaningful	Mostly meaningful	Mostly meaningless	Completely meaningless	Comments
Community robustness: «The ability of the established processes in a community to guarantee the delivery of robust products».	□	□	□	□	
1. Maturity of security process: «Degree to which a development community has established processes dedicated to guarantee the security of delivered products».	□	□	□	□	Preventive and reactive actions should be covered as subcharacteristics. Preventive would include best practices (audits...) Reactive includes procedures and reaction time.
2. Maturity of reliability process: «Degree to which a development community has established processes dedicated to guarantee that delivered products are free of critical defects (defects that prevent the operation of the product under common operation conditions)».	□	□	□	□	What does critical actually means? Critical for the developer is not necessarily critical for the user. Restructuring can be critical for the developer.

Separation of security and reliability is important. There are quite different issues in both areas.

Question 2: When evaluating the community robustness of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*			.....						*Completely irrelevant*	*Don´t Know*
1. Maturity of security process	X	9	8	7	6	5	4	3	2	1	0	
2. Maturity of reliability process	10	9	X	7	6	5	4	3	2	1	0	

Question 4: When evaluating OSS for use as platform level, how relevant are these characteristics for you? **(For each additional characteristic, ask HOW RELEVANT IT IS?)**

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Maturity of security process	X	9	8	7	6	5	4	3	2	1	0	
2. Maturity of reliability process	10	9	X	7	6	5	4	3	2	1	0	

9.8 Business Goals

The goal of this part is to elicit potential usage scenarios of the QualOSS platform, and how we can focus our work to support users in future.

Question 1: In which situations or scenarios would you like to have the support of such an evaluation? Are there any specific aspects that you find difficult to evaluate now, and where you would like to have support from QualOSS?

From interviewee 1:

You can rely on famous products and don't worry a lot about whether they work or not, or you can take less known or risky ones. For the second case you need a lot more support.

Evaluation process is not formal and difficult to formalize. Intw. would be reluctant to take an approach that just automatically gives a green light. He would like to be able to use hist judgement.

Usual approach: problem -> ask people in the company who may have experience -> use their feedback to look in internet and form an own opinion.-> download, play with the product, look at documentation.

If information is not available about some aspect, this is not a blocker. Other sources may be useful.

Usability is easy to evaluate by directly testing the product. Maintainability is harder to evaluate. Support can be done but takes time. Robustness is difficult, it is evaluated during use, it is hard to find problems in advance.

It is generally hard to find information from different sources and collect it so that it is useful. QualOSS can help there.

From interviewee 2:

At the moment rely on fame of products; downloading/"hello world". If you try to dig deeper, requires a lot of time (get to bug tracking, get sources, see whether releases are made regularly, etc.). Any support would help

Usually non-formal evaluation; going to sourceforge etc.; results need to be transparent

Judgement needs to be involved; see how the tool arrives at its conclusion

Scenario: ask colleagues for OSS component that supports specific tasks; find & download --> project lively, documentation clear enough, play with it (role: User, no integration into existing software)

usability easy, maintainability difficult; support (black/white: support or none; quality of support difficult to see), robustness difficult (using the product; or test suites)

Question 2: Now, that we discussed business goals or usage scenarios, are there any additional characteristics that come to your mind?

From interviewee 1:

Licensing as an important issue. For internal use, is not that important. For inclusion in products, it is serious: liability, importance of playing with the rules: being honest, being a good player in the community.

From interviewee 2:

Licensing is important. Permissiveness and obligation are two aspects of that --> Liability plays a role

9.9 Interview 2

9.10 Organizational Information

Date	Dec. 21st 2007
Start time	10:15 (length about 2 hours)
Company	Organization B
Context / Domain	Public organization (computing infrastructure for city representative and citizens of the city)
Interviewee Position	Director for Technology and Infrastructure

1. Do you use F/OSS components as end-application: Yes .

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

Embedded	External service	Internal service	Desktop	Development
.	X	X	X	X

2. Do you use F/OSS components as platform level (to build an application): Yes.

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

Embedded	External service	Internal service	Desktop	Development
.	X	X	X	X

3. How do you use F/OSS components

   X Integrating an F/OSS product into a company's infrastructure

   X Integrating an OSS product/components into a software product/system developed by a company

   . Forking an existing open source component

   . Extending an open source product to communicate with my product

   X Selecting an open source language and libraries to develop my product on

10 Characteristic: Community evolvability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Community Evolvability: «The likelihood that a F/OSS community remains able to maintain the product or products it develops over an extended period of time».	□	□	X	□	See comment 0 below
1. Product adoption: «Extent to which a F/OSS product is actively used by individuals and organization around the world».	X	□	□	□	
2. Developer community liveness: «Amount of work put by a development community into the creation and further development of a software product over a certain period of time».	X	□	□	□	
3. Process maturity: «Ability of a developer community to achieve development related goals by following established processes. Additionally, the level to which the processes followed by a development community are able to guarantee that certain desired product characteristics will be present in the product».	□	X	□	□	See Comment 3 below.
4. Support availability «Ease with which a user can get support (e.g., engage experienced individuals or organizations) to perform tasks that make it possible to use a product for a particular purpose».	□	X	□	□	

Comment 0. too restrictive: What about aspect related to human relationships. Motivation of community members is not tied to the product.

Comment 3. Should also include: know-how of community or its members, ability to work in team, willingness to spend time to address end user support questions

Question 2: When evaluating the community evolvability OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*
	Taste for Innovation	

	Balance of power	is the power well balanced between the different actors of the community

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Product adoption	10	9	**8**	7	6	5	4	3	2	1	0	
2. Developer community liveness	10	9	**8**	7	6	5	4	3	2	1	0	
3. Process maturity	10	9	8	7	6	5	**4**	3	2	1	0	
4. Support availability	10	9	8	7	6	5	**4**	3	2	1	0	

Question 4: When evaluating OSS for use as platform level, how relevant are these characteristics for you? ***For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Product adoption	10	9	8	7	**6**	5	4	3	2	1	0	
2. Developer community liveness	10	**9**	8	7	6	5	4	3	2	1	0	
3. Process maturity	10	9	**8**	7	6	5	4	3	2	1	0	
4. Support availability	10	**9**	8	7	6	5	4	3	2	1	0	

11 Characteristic: Community Robustness

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Community robustness: «The ability of the established processes in a community to guarantee the delivery of robust products».	□	□	X	□	Only related to product, this definition miss community aspect
1. Maturity of security process: «Degree to which a development community has established processes dedicated to guarantee the security of delivered products».	X	□	□	□	
2. Maturity of reliability process: «Degree to which a development community has established processes dedicated to guarantee that delivered products are free of critical defects (defects that prevent the operation of the product under common operation conditions)».	X	□	□	□	

Question 2: When evaluating the community robustness of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*
	Frequency of releases	This is not a characteristic per se but it is a factor that may impact robustness and also shows whether the community follows a rigorous development process.

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Maturity of security process	10	9	**8**	7	6	5	4	3	2	1	0	
2. Maturity of reliability process	10	**9**	8	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Maturity of security process	10	**9**	8	7	6	5	4	3	2	1	0	
2. Maturity of reliability process	10	9	**8**	7	6	5	4	3	2	1	0	

QUESTIONS REGARDING SUBCHARACTERISTICS WERE NOT ASKED DURING THE INTERVIEW

12 Business Goals

The goal of this part is to elicit potential usage scenarios of the QualOSS platform, and how we can focus our work to support users in future.

Question 1: In which situations or scenarios would you like to have the support of such an evaluation? Are there any specific aspects that you find difficult to evaluate now, and where you would like to have support from QualOSS?

In our case, our business goals is to increase SME services around FlOSS so that we only need to be minimally involved in development.

As a public organization, the Technology and Infrastructure Department of Sambreville does not have large amount of resources to allocate to the development of full product. Instead, we only want to get involved in the development just to know enough in case we do not find other local software development companies to perform the job. However, we feel that we must keep a small active part in development so as to stay up to speed with what is going on and know how to judge the quality of the solution proposed by local SME or in case, we do not find anyone to answer our call for tender then we can eventually develop certain modules ourselves.

12.1 Interview 3

12.2 Organizational Information

Date	
Start time	
Company	Organization A
Context / Domain	Development, integration, documentation, mentoring of new people.
Interviewee Position	Software engineer, TopCase, tool manager

1. Do you use F/OSS components as end-application: . Yes. No.

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

   X

   Embedded External service Internal service Desktop Development

2. Do you use F/OSS components as platform level (to build an application): . Yes. No.

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

   X

   Embedded External service Internal service Desktop Development

3. How do you use F/OSS components

   . Integrating an F/OSS product into a company's infrastructure

   X Integrating an OSS product/components into a software product/system developed by a company

   . Forking an existing open source component

   . Extending an open source product to communicate with my product

   . X Selecting an open source language and libraries to develop my product on

13 Characteristic: Product Evolvability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Product evolvability: "The ability of a product to be corrected, adapted and extended over time, according to the needs of its users".	X	☐	☐	☐	
1. Usefulness of code documentation: «Extent to which the source code documentation (explicitly describing the product's internals) is useful when performing corrections, adaptations or extensions to the product».	X	☐	☐	☐	This should be extended to code readability.
2. Usefulness of user documentation: «Extent to which the product's user/administrator oriented documentation is useful when deploying and using the product».	X	☐	☐	☐	
3. Maintainability: «Amount of effort required by a programmer or team of programmers with no previous knowledge of the product, to understand its code to the point that successful modifications are possible».	☐	☐	☐	☐	Maintainability is actually in inverse relation to effort involved.
4. Portability: «Ease with which a system or component can be transferred from one hardware or software environment to another».	X	☐	☐	☐	
5. Interoperability: «Degree to which a software product can interoperate with other software product either live or based on input/output data ».	☐	☐	☐	☐	It is not clear what is meant by interoperate: clarify what it means.
6. Compliance to standards: «Degree to which a product complies with published standards that are relevant to its functionality. Important note: for measurement purposes, this criterion is applied separately to various relevant software artefacts, i.e., source code, documentation, etc»	☐	☐	☐	☐	It may be important to distinguish between de-facto standards and norms.

Question 2: When evaluating the product evolvability of OSS components, would you consider any additional characteristic?

No.	*Characteristic*	*Description*
	Ease of deployment Easy of distribution	Software dependencies, ease of packaging for different platforms.

	(Not really related to evolvability or robustness)	
	Availability of information in the internet	Blogs, mailing lists, forums

Question 3: When evaluating OSS for use as end application, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Usefulness of code documentation	10	9	8	X	6	5	4	3	2	1	0	
2. Usefulness of user documentation	10	X	8	7	6	5	4	3	2	1	0	
3. Maintainability	10	X	8	7	6	5	4	3	2	1	0	
4. Portability	10	9	X	7	6	5	4	3	2	1	0	
5. Interoperability	10	9	8	7	6	5	4	3	X	1	0	
6. Compliance to standards	10	X	8	7	6	5	4	3	2	1	0	
Availability of information in the internet	10	X	8	7	6	5	4	3	2	1	0	

Question 4: When evaluating OSS for platform level, how relevant are these characteristics for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Usefulness of code documentation	10	X	8	7	6	5	4	3	2	1	0	
2. Usefulness of user documentation	10	9	8	7	6	X	4	3	2	1	0	This could include unofficial sources such as blogs and mailing lists.
3. Maintainability	10	X	8	7	6	5	4	3	2	1	0	
4. Portability	X	9	8	7	6	5	4	3	2	1	0	
5. Interoperability	10	9	8	7	6	5	4	3	2	1	0	X
6. Compliance to standards	10	9	X	7	6	5	4	3	2	1	0	
Availability of information in the internet	X	9	8	7	6	5	4	3	2	1	0	

14 Characteristic: Community evolvability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Community Evolvability: «The likelihood that a F/OSS community remains able to maintain the product or products it develops over an extended period of time».	X	□	□	□	
1. Product adoption: «Extent to which a F/OSS product is actively used by individuals and organization around the world».	X	□	□	□	
2. Developer community liveness: «Amount of work put by a development community into the creation and further development of a software product over a certain period of time».	X	□	□	□	
3. Process maturity: «Ability of a developer community to achieve development related goals by following established processes. Additionally, the level to which the processes followed by a development community are able to guarantee that certain desired product characteristics will be present in the product».	X	□	□	□	
4. Support availability «Ease with which a user can get support (e.g., engage experienced individuals or organizations) to perform tasks that make it possible to use a product for a particular purpose».	□	X	□	□	It should be clarified which type of support this is: community, commercial, etc.

Question 2: When evaluating the community evolvability OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*	.....								*Completely irrelevant*	*Don´t Know*
1. Product adoption	10	X	8	7	6	5	4	3	2	1	0	
2. Developer community liveness	10	9	X	7	6	5	4	3	2	1	0	

3. Process maturity	10	9	8	7	6	5	4	X	2	1	0	*
4. Support availability	10	9	8	7	6	5	4	3	X	1	0	*

- *Focused on building infrastructure.*

Question 4: When evaluating OSS for use as platform level, how relevant are these characteristics for you? ***For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant* ..... *Completely irrelevant*											*Don´t Know*
1. Product adoption	10	X	8	7	6	5	4	3	2	1	0	
2. Developer community liveness	10	9	X	7	6	5	4	3	2	1	0	
3. Process maturity	10	9	8	7	X	5	4	3	2	1	0	
4. Support availability	10	9	8	7	6	5	X	3	2	1	0	

15 Characteristic: Product Robustness

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Product Robustness: «Degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions».	X	□	□	□	
1. Reliability: «Ability of a system or component to perform its required functions under stated conditions for a specified period of time».	X	□	□	□	
2. Security: « Capability of the software product to protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them».	X	□	□	□	
3. Maturity: «Degree to which the general, long term objectives set for a product have been reached by current implementation».	X	□	□	□	

Question 2: When evaluating the product robustness of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*			.....						*Completely irrelevant*	*Don´t Know*
1. Reliability	10	9	8	X	6	5	4	3	2	1	0	
2. Security	10	X	8	7	6	5	4	3	2	1	0	
3. Maturity	10	9	X	7	6	5	4	3	2	1	0	

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely*	*Completely*	*Don´t Know*

	relevant					.....				*irrelevant*		
1. Reliability	10	X	8	7	6	5	4	3	2	1	0	
2. Security	10	X	8	7	6	5	4	3	2	1	0	
3. Maturity	10	X	8	7	6	5	4	3	2	1	0	

16 Characteristic: Community Robustness

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Community robustness: «The ability of the established processes in a community to guarantee the delivery of robust products».	X	□	□	□	
1. Maturity of security process: «Degree to which a development community has established processes dedicated to guarantee the security of delivered products».	X	□	□	□	
2. Maturity of reliability process: «Degree to which a development community has established processes dedicated to guarantee that delivered products are free of critical defects (defects that prevent the operation of the product under common operation conditions)».	X	□	□	□	Reliability sounds too general, security is specific. Restructuring may make sense here, e.g., promoting some of the subcharacteristics of reliability.

Question 2: When evaluating the community robustness of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*

Question 3: When evaluating OSS for use as end application, how relevant are these characteristic for you? ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*			.....						*Completely irrelevant*	*Don´t Know*
1. Maturity of security process	10	9	8	7	6	5	4	3	2	1	X	*
2. Maturity of reliability process	10	9	8	7	6	5	4	3	2	1	X	*

** Just not considered for analysis.*

Question 4: *When evaluating OSS for use as platform level, how relevant are these characteristics for you?* ***(For each additional characteristic, ask HOW RELEVANT IT IS?)***

Sub-characteristic	*Completely relevant*				.....					*Completely irrelevant*		*Don´t Know*
1. Maturity of security process	10	9	8	7	6	5	4	3	2	1	X	*1
2. Maturity of reliability process	10	9	8	7	6	X	4	3	2	1	0	*2

**1 Not considered for analysis*

**2 Many aspects considered: release process, stable/unstable branches well defined (what goes into which one), regular developer 'meetings' (chat...).*

17 Business Goals

The goal of this part is to elicit potential usage scenarios of the QualOSS platform, and how we can focus our work to support users in future.

Question 1: In which situations or scenarios would you like to have the support of such an evaluation? Are there any specific aspects that you find difficult to evaluate now, and where you would like to have support from QualOSS?

Scenario 1: Find a wiki system for internal use in the company. Used an existing comparison page. Looked at date of last modification and drop static projects. Desire of being able to maintain the software in-house, so looked at the implementation (language). Search for certain characteristics (interoperability) and filter. Then look for user adoption (reviews, etc). Test two implementations by deploying them.

Better indicators of community liveliness would have been better (only date of last modif. used). Also indicators of the maturity of the technology overall. For example, date of first release.

Scenario 2: Search for an appropriate graphical toolkit library. Looked for a reach toolkit: options available: Gtk and Qt. Gtk selected: It it easier to create bindings to C than to C++. Strong community.

QualOSS can help with support for community evaluation.

Reliability issues were hard to see before doing the work of integrating the library.

Question 2: Now, that we discussed business goals or usage scenarios, are there any additional characteristics that come to your mind?

Ease of distribution. If your product depends on FLOSS components, their ease of distribution and deployment is important for the final users.

Licensing is central. It is probably one of the first points you look at.

17.1 Interview 4

17.2 Organizational Information

Date	28/11/2007
Start time	
Company	Organization C
Context / Domain	Software publisher for public sector

1. Do you use F/OSS components as end-application:. **Yes**. No.

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

.	.X	.X	.	.X
Embedded	External service	Internal service	Desktop	Development

2. Do you use F/OSS components as platform level (to build an application):. **Yes**. No.

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

.	.X	.X	.	.X
Embedded	External service	Internal service	Desktop	Development

3. How do you use F/OSS components

   .X Integrating an F/OSS product into a company's infrastructure

   .X Integrating an OSS product/components into a software product/system developed by a company

   . Forking an existing open source component

   . Extending an open source product to communicate with my product

   .X Selecting an open source language and libraries to develop my product on

1Characteristic: Evolvability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Evolvability: «General ability of a F/OSS project to deliver useful products (or product updates) over an extended period of time. Also the ability of such products to remain useful for an extended period of time».	X	□	□	□
2. Product evolvability: «Ability of a product to be corrected, adapted and extended over time, according to the needs of its users».	X	□	□	□
3. Community evolvability: «Likelihood that a F/OSS community remains able to maintain the product or products it develops over an extended period of time».	X	□	□	□

Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant:

Question 2: How relevant do you thing the following characteristic are to asses the quality of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Evolvability	10	9	(8)	7	6	5	4	3	2	1	0
2. Product evolvability	10	9	(8)	7	6	5	4	3	2	1	0
3. Community evolvability	10	9	8	7	(6)	5	4	3	2	1	0

Question 3: How relevant do you thing the following characteristic are to asses the quality of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Evolvability	10	9	(8)	7	6	5	4	3	2	1	0
2. Product evolvability	(10)	9	8	7	6	5	4	3	2	1	0
3. Community evolvability	10	9	(8)	7	6	5	4	3	2	1	0

17.3 2Characteristic: Product Evolvability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Usefulness of code documentation: «Extent to which the source code documentation (explicitly describing the product's internals) is useful when performing corrections, adaptations or extensions to the product».	□	□	X	□
2. Usefulness of user documentation: «Extent to which the product's user/administrator oriented documentation is useful when deploying and using the product».	X	□	□	□
3. Maintainability: «Amount of effort required by a programmer or team of programmers with no previous knowledge of the product, to understand its code to the point that successful modifications are possible».	□	X	□	□
4. Portability: «Ease with which a system or component can be transferred from one hardware or software environment to another».	X	□	□	□
5. Interoperability: «Degree to which a software product can interoperate with other software product either live or based on input/output data ».	X	□	□	□
6. Compliance to standards: «Degree to which a product complies with published standards that are relevant to its functionality. Important note: for measurement purposes, this criterion is applied separately to various relevant software artefacts, i.e., source code, documentation, etc»	□	X	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the product evolvability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Usefulness of code documentation	10	9	(8)	7	6	5	4	3	2	1	0
2. Usefulness of user documentation	(10)	9	8	7	6	5	4	3	2	1	0
3. Maintainability	10	9	(8)	7	6	5	4	3	2	1	0
4. Portability	(10)	9	8	7	6	5	4	3	2	1	0

5. Interoperability	10	9	(8)	7	6	5	4	3	2	1	0
6. Compliance to standards	10	9	8	(7)	6	5	4	3	2	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the product evolvability?

Large community use

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the product evolvability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Usefulness of code documentation	10	9	(8)	7	6	5	4	3	2	1	0
2. Usefulness of user documentation	10	9	(8)	7	6	5	4	3	2	1	0
3. Maintainability	(10)	9	8	7	6	5	4	3	2	1	0
4. Portability	10	9	8	7	6	(5)	4	3	2	1	0
5. Interoperability	(10)	9	8	7	6	5	4	3	2	1	0
6. Compliance to standards	10	9	(8)	7	6	5	4	3	2	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the product evolvability?

Large community use

17.3.1 2.1Characteristic: Usefulness of code documentation

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Actuality: «Extent to which the code documentation describes the current version of the source code as opposite to describing older versions of it»**.**	□	X	□	□
2. Coverage: «Ratio between size of documented code and general product code size»	□	□	□	X
3. Code documentation standard compliance: «Degree to which a product complies with published standards that are relevant to its functionality »	□	□	X	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses usefulness of the code documentation of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Actuality	10	9	8	7	6	(5)	4	3	2	1	0
2. Coverage	10	9	8	7	6	(5)	4	3	2	1	0
3. Code documentation standard compliance	10	9	8	7	6	(5)	4	3	2	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the usefulness of the code documentation?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses usefulness of the code documentation of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Actuality	10	9	8	7	6	(5)	4	3	2	1	0
2. Coverage	10	9	8	7	6	(5)	4	3	2	1	0
3. Code documentation standard compliance	10	9	8	7	6	(5)	4	3	2	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the usefulness of the code documentation?

17.3.2 2.2Characteristic: Usefulness of user documentation

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Actuality: «Extent to which the user documentation describes the current version of the product functionality as opposite to describing outdated functionality»..	X	□	□	□
2. Coverage: «Ratio between the number of documented product features and the general number of features offered by the product»	X	□	□	□
3. Internationalization: «Availability of the documentation in various natural languages»	□	X	□	□
4. User documentation standard compliance: «Degree to which a product complies with published standards relevant to documentation»	□	□	□	X

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses usefulness of the user documentation of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Actuality	(10)	9	8	7	6	5	4	3	2	1	0
2. Coverage	(10)	9	8	7	6	5	4	3	2	1	0
3. Internationalization	10	9	(8)	7	6	5	4	3	2	1	0
4. User documentation standard compliance	10	9	8	7	6	5	(4)	3	2	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the usefulness of the user documentation?

Online with contextual and extended search capabilities

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses usefulness of the user documentation of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Actuality	(10)	9	8	7	6	5	4	3	2	1	0
2. Coverage	(10)	9	8	7	6	5	4	3	2	1	0
3. Internationalization	10	9	8	7	6	(5)	4	3	2	1	0
4. User documentation standard compliance	10	9	8	7	6	5	4	3	2	(1)	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the usefulness of the user documentation?

17.3.3 2.3Characteristic: Maintainability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Product complexity: «Degree which system or component has a design or implementation that is difficult to understand and verify»	□	X	□	□
2. Architecture flexibility: «Ability of the product's architecture of being applied to new problems. The ease which with a system or component can be modified for use in applications or environments other than those for which it was specifically designed»	X	□	□	□
3. Fix ability: «Ease with which a software product can be fixed»	□	X	□	□
4. Maintainability standard compliance: «Degree to which a product complies with published standards relevant to maintainability»	□	□	X	□
5. Product buildability: «Degree to which a system or component can be rebuild after modifications to the source»	□	X	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses maintainability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Product complexity	10	9	(8)	7	6	5	4	3	2	1	0
2. Architecture flexibility	(10)	9	8	7	6	5	4	3	2	1	0
3. Fix ability:	10	9	(8)	7	6	5	4	3	2	1	0
4. Maintainability standard compliance	10	9	8	7	6	5	4	(3)	2	1	0
5. Product buildability	10	9	8	(7)	6	5	4	3	2	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the maintainability?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses maintainability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Product complexity	10	9	(8)	7	6	5	4	3	2	1	0
2. Architecture flexibility	10	9	(8)	7	6	5	4	3	2	1	0
3. Fix ability:	10	9	(8)	7	6	5	4	3	2	1	0
4. Maintainability standard compliance	10	9	8	7	6	5	4	3	(2)	1	0
5. Product buildability	10	9	(8)	7	6	5	4	3	2	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the maintainability?

17.3.4 2.4Characteristic: Portability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Platform specificity: «Degree to which a product's code is specific to a particular hardware or software environment»	X	□	□	□
2. Standard compliance: «Degree to which a product complies with published standards that are relevant to its functionality»	□	□	X	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses portability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Product specificity	(10)	9	8	7	6	5	4	3	2	1	0
2. Standard compliance	10	9	8	7	6	(5)	4	3	2	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the portability?

Development language used

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses portability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Product specificity	(10)	9	8	7	6	5	4	3	2	1	0
2. Standard compliance	10	9	8	7	6	(5)	4	3	2	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the portability?

17.3.5 2.5Characteristic: Interoperability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Runtime interoperability: «Interoperability with other software products while in operation»	□	X	□	□
2. Passive interoperability: «Interoperability with other software products based on output data generated by the software product or based on the capacity of the software product to read various data types and formats»	□	X	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses interoperability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Runtime interoperability	10	9	(8)	7	6	5	4	3	2	1	0
2. Passive interoperability	10	9	(8)	7	6	5	4	3	2	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the interoperability?

Standard compliance

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses interoperability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Runtime interoperability	10	9	(8)	7	6	5	4	3	2	1	0
2. Passive interoperability	10	9	(8)	7	6	5	4	3	2	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the interoperability?

17.4 3Characteristic: Community evolvability

Question 1 : How meaningful are these definitions for you ? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Product adoption: «Extent to which a F/OSS product is actively used by individuals and organization around the world».	X	□	□	□
2. Developer community liveness: «Amount of work put by a development community into the creation and further development of a software product over a certain period of time».	□	X	□	□
3. Process maturity: «Ability of a developer community to achieve development related goals by following established processes. Additionally, the level to which the processes followed by a development community are able to guarantee that certain desired product characteristics will be present in the product».	□	X	□	□
4. Support availability «Ease with which a user can engage experienced individuals or organizations (on a for-profit or voluntary basis) to perform tasks that make it possible to use a product for a particular purpose».	□	X	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the community evolvability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Product adoption	10	9	(8)	7	6	5	4	3	2	1	0
2. Developer community liveness	10	9	(8)	7	6	5	4	3	2	1	0
3. Process maturity	10	9	(8)	7	6	5	4	3	2	1	0
4. Support availability	10	9	(8)	7	6	5	4	3	2	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the community evolvability?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the community evolvability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Product adoption	10	9	(8)	7	6	5	4	3	2	1	0
2. Developer community liveness	10	9	(8)	7	6	5	4	3	2	1	0
3. Process maturity	10	9	(8)	7	6	5	4	3	2	1	0
4. Support availability	10	9	(8)	7	6	5	4	3	2	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the community evolvability?

17.4.1 3.1Characteristic: Product adoption

Question 1 : How meaningful are these definitions for you ? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. User community size: «Number of users (individuals or organizations) that use a F/OSS product worldwide»**.**	□	X	□	□
2. Mission criticality: «Extent to which users of a product apply it to mission-critical tasks. Alternatively, the degree to which users of a product depend on the product for reaching their business goals»	□	□	X	□
3. License permissiveness: «Amount of freedom allowed to product users by the product's licence»	X	□	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses product adoption of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. User community size	10	9	(8)	7	6	5	4	3	2	1	0
2. Mission criticality	10	9	8	7	6	(5)	4	3	2	1	0
3. License permissiveness	(10)	9	8	7	6	5	4	3	2	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses product adoption?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses product adoption of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. User community size	10	9	(8)	7	6	5	4	3	2	1	0
2. Mission criticality	10	9	(8)	7	6	5	4	3	2	1	0
3. License permissiveness	10	9	8	7	(6)	5	4	3	2	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses product adoption?

17.4.2 3.2Characteristic: Developer community liveness

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Developer community size: «Number of individuals and organizations actively contributing to a product's development over a certain period of time»**.**	□	X	□	□
2. Developer community activity: «General number and size of the contributions made to a product's development over a certain period of time»	□	□	X	□
3. Developer community heterogeneity: «Degree to which different types of developers (e.g., individuals vs. organizations, for-profit vs. non-for-profit, hobbyist vs paid professionals) are present in a developer community»	□	□	□	X
4. Developer community fluctuation: «Rate movement of people into, and out of a developer community over time»	□	□	□	X

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses developer community liveness of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Developer community size	10	9	(8)	7	6	5	4	3	2	1	0
2. Developer community activity	10	9	8	7	6	(5)	4	3	2	1	0
3. Developer community heterogeneity	10	9	8	7	6	5	4	3	(2)	1	0
4. Developer community fluctuation	10	9	8	7	6	5	4	3	(2)	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses developer community liveness ?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses developer community liveness of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Developer community size	10	9	(8)	7	6	5	4	3	2	1	0

2. Developer community activity	10	9	8	7	6	(5)	4	3	2	1	0
3. Developer community heterogeneity	10	9	8	7	6	5	4	3	(2)	1	0
4. Developer community fluctuation	10	9	8	7	6	5	4	3	(2)	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses developer community liveness ?

17.4.3 3.3Characteristic: Process maturity

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Established process coverage: «Degree to which the development activities a community performs are covered by established, repeatable processes that are widely known and accepted by community members. Development processes that have been observed to be well established in existing development communities include project management, quality assurance and requirement engineering»**.**	□	X	□	□
2. Process automation: «Degree to which established processes are partially or completely automated though the use of software tools. Examples of software tools commonly used by development communities to automate software processes include bug tracking systems, build farms and build daemons, and automated test suites»	□	□	X	□
3. Popularization: « Availability of support related to popularize a software product ».	□	X	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses process maturity of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Established process coverage	10	9	(8)	7	6	5	4	3	2	1	0
2. Process automation	10	9	8	7	6	5	(4)	3	2	1	0
3. Popularization	10	9	(8)	7	6	5	4	3	2	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses process maturity?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses process maturity of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Established process coverage	10	9	(8)	7	6	5	4	3	2	1	0
2. Process automation	10	9	8	7	6	(5)	4	3	2	1	0
3. Popularization	10	9	(8)	7	6	5	4	3	2	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses process maturity?

17.4.4 3.4Characteristic: Support availability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Modification support availability: «Availability of support related to performing specific modifications to a software product»**.**	X	□	□	□
2. Deployment support: «Availability of support related to solving problems arising from the deployment and use of a software product»	□	X	□	□
3. Backward support: «Availability of support related to older version of a software product still in use»	□	□	□	X

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses support availability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Modification support availability	(10)	9	8	7	6	5	4	3	2	1	0
2. Deployment support	10	9	(8)	7	6	5	4	3	2	1	0
3. Backward support	10	9	8	7	6	5	4	3	(2)	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses support availability?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses support availability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Modification support availability	10	9	(8)	7	6	5	4	3	2	1	0
2. Deployment support	10	9	8	7	6	(5)	4	3	2	1	0
3. Backward support	10	9	8	7	6	5	4	(3)	2	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses support availability?

17.5 4Characteristic: Robustness

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaningless?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Robustness: «General ability of a F/OSS project to deliver robust products over an extended period of time.».	X	□	□	□
2. Product robustness: «Degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions ».	X	□	□	□
3. Community robustness: «Ability of the established processes in a community to guarantee the delivery of robust products».	□	□	X	□

Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant:

Question 2: How relevant do you thing the following characteristic are to asses the quality of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Robustness	(10)	9	8	7	6	5	4	3	2	1	0
2. Product robustness	(10)	9	8	7	6	5	4	3	2	1	0
3. Community robustness	10	9	8	7	6	5	(4)	3	2	1	0

Question 3: How relevant do you thing the following characteristic are to asses the quality of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Robustness	(10)	9	8	7	6	5	4	3	2	1	0
2. Product robustness	(10)	9	8	7	6	5	4	3	2	1	0
3. Community robustness	10	9	8	7	6	(5)	4	3	2	1	0

17.6 5Characteristic: Product Robustness

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Reliability: «Ability of a system or component to perform its required functions under stated conditions for a specified period of time».	X	□	□	□
2. Security: *«* Capability of the software product to protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them*»*.	X	□	□	□
3. Maturity: «Degree to which the general, long term objectives set for a product have been reached by current implementation».	□	X	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the product robustness of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Reliability	(10)	9	8	7	6	5	4	3	2	1	0
2. Security	(10)	9	8	7	6	5	4	3	2	1	0
3. Maturity	10	9	(8)	7	6	5	4	3	2	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the product robustness?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the product robustness of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Reliability	(10)	9	8	7	6	5	4	3	2	1	0
2. Security	(10)	9	8	7	6	5	4	3	2	1	0
3. Maturity	10	9	8	(7)	6	5	4	3	2	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the product robustness?

17.6.1 5.1Characteristic: Reliability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Failure tolerance: «Capability of the software product to avoid failure as a result of faults in the software»**.**	□	X	□	□
2. Fault tolerance: «Capability of the software product to maintain a specified level of performance in cases of software faults or of infringement of its specified interface»	□	□	X	□
3. Recoverability: «Capability of the software product to re-establish a specified level of performance and recover the data directly affected in the case of a failure»	□	X	□	□
4. Availability: «Degree to which a system or component is operational and accessible when required for use»	□	X	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses reliability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Failure tolerance	10	9	(8)	7	6	5	4	3	2	1	0
2. Fault tolerance	10	9	8	7	6	(5)	4	3	2	1	0
3. Recoverability	10	9	(8)	7	6	5	4	3	2	1	0
4. Availability	10	9	(8)	7	6	5	4	3	2	1	0

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the reliability?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses reliability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Failure tolerance	10	9	(8)	7	6	5	4	3	2	1	0
2. Fault tolerance	10	9	8	7	6	(5)	4	3	2	1	0
3. Recoverability	10	9	(8)	7	6	5	4	3	2	1	0
4. Availability	10	9	(8)	7	6	5	4	3	2	1	0

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the reliability?

17.6.2 5.2Characteristic: Security

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Confidentiality: «Degree to which a system prevents unauthorized disclosure of information; that is, provides assurance that information is not disclosed to unauthorized individuals, processes, or devices»**.**	X	□	□	□
2. Integrity: «Degree to which a system or component is able to protect the accuracy and completeness of information and processing methods»	X	□	□	□
3. Security standards compliance: «Degree to which a product complies with published security standards that are relevant to its functionality»	□	□	X	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses security of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Confidentiality	(10)	9	8	7	6	5	4	3	2	1	0
2. Integrity	(10)	9	8	7	6	5	4	3	2	1	0
3. Security standards compliance	10	9	8	7	6	(5)	4	3	2	1	0

17.7 Interview 5

17.8 Organizational Information

Date	27-11-2007
Start time	
Company	Organization D
Context / Domain	E-gouvernement
Interviewee Position	IT expert

1. Do you use F/OSS components as end-application: & Yes. No.

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

.	&	&	&	&
Embedded	External service	Internal service	Desktop	Development

2. Do you use F/OSS components as platform level (to build an application): & Yes. No.

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

.	&	&	.	&
Embedded	External service	Internal service	Desktop	Development

3. How do you use F/OSS components

   & Integrating an F/OSS product into a company's infrastructure

   & Integrating an OSS product/components into a software product/system developed by a company

   & Forking an existing open source component

   . Extending an open source product to communicate with my product

   & Selecting an open source language and libraries to develop my product on

17.9 Characteristic: Evolvability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Evolvability: «General ability of a F/OSS project to deliver useful products (or product updates) over an extended period of time. Also the ability of such products to remain useful for an extended period of time».	&	□	□	□
2. Product evolvability: «Ability of a product to be corrected, adapted and extended over time, according to the needs of its users».	□	&	□	□
3. Community evolvability: «Likelihood that a F/OSS community remains able to maintain the product or products it develops over an extended period of time».	□	&	□	□

Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant:

Question 2: How relevant do you thing the following characteristic are to asses the quality of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Evolvability	10										
2. Product evolvability		9									
3. Community evolvability			8								

Question 3: How relevant do you thing the following characteristic are to asses the quality of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Evolvability	10										
2. Product evolvability		9									
3. Community evolvability			8								

17.10 Characteristic: Product Evolvability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Usefulness of code documentation: «Extent to which the source code documentation (explicitly describing the product's internals) is useful when performing corrections, adaptations or extensions to the product».	&	□	□	□
2. Usefulness of user documentation: «Extent to which the product's user/administrator oriented documentation is useful when deploying and using the product».	&	□	□	□
3. Maintainability: «Amount of effort required by a programmer or team of programmers with no previous knowledge of the product, to understand its code to the point that successful modifications are possible».	□	&	□	□
4. Portability: «Ease with which a system or component can be transferred from one hardware or software environment to another».	□	&	□	□
5. Interoperability: «Degree to which a software product can interoperate with other software product either live or based on input/output data ».	□	&	□	□
6. Compliance to standards: «Degree to which a product complies with published standards that are relevant to its functionality. Important note: for measurement purposes, this criterion is applied separately to various relevant software artefacts, i.e., source code, documentation, etc»	&	□	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the product evolvability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Usefulness of code documentation			8								
2. Usefulness of user documentation	10										
3. Maintainability			8								
4. Portability						5					
5. Interoperability					6						
6. Compliance to standards			8								

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the product evolvability?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the product evolvability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Usefulness of code documentation					6						
2. Usefulness of user documentation	10										
3. Maintainability			8								
4. Portability			8								
5. Interoperability			8								
6. Compliance to standards	10										

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the product evolvability?

17.10.1 Characteristic: Usefulness of code documentation

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Actuality: «Extent to which the code documentation describes the current version of the source code as opposite to describing older versions of it»**.**	□	&	□	□
2. Coverage: «Ratio between size of documented code and general product code size»	□	&	□	□
3. Code documentation standard compliance: «Degree to which a product complies with published standards that are relevant to its functionality »	□	&	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses usefulness of the code documentation of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Actuality				7							
2. Coverage				7							
3. Code documentation standard compliance					6						

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the usefulness of the code documentation?

_____

_____

_____

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses usefulness of the code documentation of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Actuality					6						
2. Coverage				7							
3. Code documentation standard compliance					6						

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the usefulness of the code documentation?

17.10.2 Characteristic: Usefulness of user documentation

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Actuality: «Extent to which the user documentation describes the current version of the product functionality as opposite to describing outdated functionality»..	&	□	□	□
2. Coverage: «Ratio between the number of documented product features and the general number of features offered by the product»	&	□	□	□
3. Internationalization: «Availability of the documentation in various natural languages»	□	&	□	□
4. User documentation standard compliance: «Degree to which a product complies with published standards relevant to documentation»	□	□	&	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses usefulness of the user documentation of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Actuality	10										
2. Coverage	10										
3. Internationalization						5					
4. User documentation standard compliance									2		

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the usefulness of the user documentation?

_____

_____

_____

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses usefulness of the user documentation of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Actuality		9									

2. Coverage		9									
3. Internationalization						5					
4. User documentation standard compliance										1	

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the usefulness of the user documentation?

17.10.3 Characteristic: Maintainability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Product complexity: «Degree which system or component has a design or implementation that is difficult to understand and verify»	&	□	□	□
2. Architecture flexibility: «Ability of the product's architecture of being applied to new problems. The ease which with a system or component can be modified for use in applications or environments other than those for which it was specifically designed»	□	&	□	□
3. Fix ability: «Ease with which a software product can be fixed»	□	&	□	□
4. Maintainability standard compliance: «Degree to which a product complies with published standards relevant to maintainability»	□	□	&	□
5. Product buildability: «Degree to which a system or component can be rebuild after modifications to the source»	□	&	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses maintainability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Product complexity	10										
2. Architecture flexibility				7							
3. Fix ability:			8								
4. Maintainability standard compliance								3			
5. Product buildability				7							

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the maintainability?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses maintainability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Product complexity			8								
2. Architecture flexibility			8								
3. Fix ability:				7							
4. Maintainability standard compliance								3			
5. Product buildability					6						

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the maintainability?

17.10.4 Characteristic: Portability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Platform specificity: «Degree to which a product's code is specific to a particular hardware or software environment»	□	□	&	□
2. Standard compliance: «Degree to which a product complies with published standards that are relevant to its functionality»	&	□	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses portability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Product specificity								3			
2. Standard compliance	10										

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the portability?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses portability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Product specificity							4				
2. Standard compliance		9									

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the portability?

17.10.5 Characteristic: Interoperability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Runtime interoperability: «Interoperability with other software products while in operation»	□	&	□	□
2. Passive interoperability: «Interoperability with other software products based on output data generated by the software product or based on the capacity of the software product to read various data types and formats»	&	□	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses interoperability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Runtime interoperability				7							
2. Passive interoperability	10										

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the interoperability?

_____

_____

_____

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses interoperability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Runtime interoperability				7							
2. Passive interoperability	10										

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the interoperability?

_____

_____

_____

17.11 Characteristic: Community evolvability

Question 1 : How meaningful are these definitions for you ? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Product adoption: «Extent to which a F/OSS product is actively used by individuals and organization around the world».	&	□	□	□
2. Developer community liveness: «Amount of work put by a development community into the creation and further development of a software product over a certain period of time».	&	□	□	□
3. Process maturity: «Ability of a developer community to achieve development related goals by following established processes. Additionally, the level to which the processes followed by a development community are able to guarantee that certain desired product characteristics will be present in the product».	□	&	□	□
4. Support availability «Ease with which a user can engage experienced individuals or organizations (on a for-profit or voluntary basis) to perform tasks that make it possible to use a product for a particular purpose».	&	□	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the community evolvability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Product adoption		9									
2. Developer community liveness	10										
3. Process maturity				7							
4. Support availability	10										

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the community evolvability?

_____

_____

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the community evolvability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*

1. Product adoption		9									
2. Developer community liveness	10										
3. Process maturity				7							
4. Support availability	10										

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the community evolvability?

17.11.1 Characteristic: Product adoption

Question 1 : How meaningful are these definitions for you ? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. User community size: «Number of users (individuals or organizations) that use a F/OSS product worldwide»**.**	&	□	□	□
2. Mission criticality: «Extent to which users of a product apply it to mission-critical tasks. Alternatively, the degree to which users of a product depend on the product for reaching their business goals»	□	&	□	□
3. License permissiveness: «Amount of freedom allowed to product users by the product's licence»	□	&	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses product adoption of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. User community size	10										
2. Mission criticality				7							
3. License permissiveness				7							

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses product adoption?

_____

_____

_____

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses product adoption of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. User community size	10										
2. Mission criticality			8								
3. License permissiveness					6						

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses product adoption?

17.11.2 Characteristic: Developer community liveness

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Developer community size: «Number of individuals and organizations actively contributing to a product's development over a certain period of time»**.**	&	□	□	□
2. Developer community activity: «General number and size of the contributions made to a product's development over a certain period of time»	&	□	□	□
3. Developer community heterogeneity: «Degree to which different types of developers (e.g., individuals vs. organizations, for-profit vs. non-for-profit, hobbyist vs paid professionals) are present in a developer community»	□	□	&	□
4. Developer community fluctuation: «Rate movement of people into, and out of a developer community over time»	□	□	&	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses developer community liveness of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Developer community size			8								
2. Developer community activity	10										
3. Developer community heterogeneity							4				
4. Developer community fluctuation							4				

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses developer community liveness ?

_____

_____

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses developer community liveness of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Developer community size		9									

2. Developer community activity	10										
3. Developer community heterogeneity							4				
4. Developer community fluctuation							4				

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses developer community liveness ?

17.11.3 Characteristic: Process maturity

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Established process coverage: «Degree to which the development activities a community performs are covered by established, repeatable processes that are widely known and accepted by community members. Development processes that have been observed to be well established in existing development communities include project management, quality assurance and requirement engineering»**.**	□	&	□	□
2. Process automation: «Degree to which established processes are partially or completely automated though the use of software tools. Examples of software tools commonly used by development communities to automate software processes include bug tracking systems, build farms and build daemons, and automated test suites»	&	□	□	□
3. Popularization: « Availability of support related to popularize a software product ».	&	□	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses process maturity of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Established process coverage				7							
2. Process automation			8								
3. Popularization		9									

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses process maturity?

_____

_____

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses process maturity of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Established process coverage				7							
2. Process automation			8								

3. Popularization		9									

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses process maturity?

17.11.4 Characteristic: Support availability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Modification support availability: «Availability of support related to performing specific modifications to a software product»**.**	&	□	□	□
2. Deployment support: «Availability of support related to solving problems arising from the deployment and use of a software product»	&	□	□	□
3. Backward support: «Availability of support related to older version of a software product still in use»	□	&	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses support availability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Modification support availability	10										
2. Deployment support		9									
3. Backward support				7							

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses support availability?

_____

_____

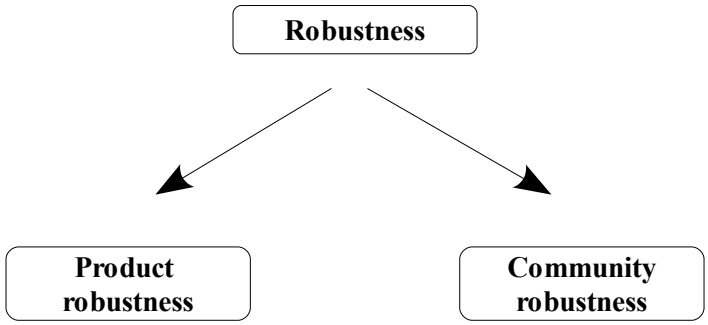
_____

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses support availability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Modification support availability				7							
2. Deployment support			8								
3. Backward support	10										

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses support availability?

17.12 Characteristic: Robustness



Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaningless?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Robustness: «General ability of a F/OSS project to deliver robust products over an extended period of time.».	&	□	□	□
2. Product robustness: «Degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions ».	□	&	□	□
3. Community robustness: «Ability of the established processes in a community to guarantee the delivery of robust products».	□	&	□	□

Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant:

Question 2: How relevant do you thing the following characteristic are to asses the quality of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Robustness		9									
2. Product robustness				7							
3. Community robustness				7							

Question 3: How relevant do you thing the following characteristic are to asses the quality of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Robustness	10										

2. Product robustness					6							
3. Community robustness			8									

17.13 Characteristic: Product Robustness

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Reliability: «Ability of a system or component to perform its required functions under stated conditions for a specified period of time».	&	□	□	□
2. Security: « Capability of the software product to protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them».	□	&	□	□
3. Maturity: «Degree to which the general, long term objectives set for a product have been reached by current implementation».	□	□	&	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the product robustness of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Reliability		9									
2. Security						5					
3. Maturity									2		

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the product robustness?

_____

_____

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the product robustness of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Reliability		9									
2. Security					6						
3. Maturity							4				

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the product robustness?

17.13.1 Characteristic: Reliability

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Failure tolerance: «Capability of the software product to avoid failure as a result of faults in the software»**.**	&	□	□	□
2. Fault tolerance: «Capability of the software product to maintain a specified level of performance in cases of software faults or of infringement of its specified interface»	&	□	□	□
3. Recoverability: «Capability of the software product to re-establish a specified level of performance and recover the data directly affected in the case of a failure»	&	□	□	□
4. Availability: «Degree to which a system or component is operational and accessible when required for use»	&	□	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses reliability of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Failure tolerance	10										
2. Fault tolerance	10										
3. Recoverability	10										
4. Availability	10										

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the reliability?

_____

_____

_____

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses reliability of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Failure tolerance		9									
2. Fault tolerance		9									

3. Recoverability		9									
4. Availability		9									

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the reliability?

17.13.2 Characteristic: Security

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Confidentiality: «Degree to which a system prevents unauthorized disclosure of information; that is, provides assurance that information is not disclosed to unauthorized individuals, processes, or devices»**.**	&	□	□	□
2. Integrity: «Degree to which a system or component is able to protect the accuracy and completeness of information and processing methods»	□	&	□	□
3. Security standards compliance: «Degree to which a product complies with published security standards that are relevant to its functionality»	□	&	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses security of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Confidentiality	10										
2. Integrity			8								
3. Security standards compliance			8								

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the security?

_____

_____

_____

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses security of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Confidentiality	10										
2. Integrity	10										
3. Security standards compliance						5					

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the security?

17.13.3 Characteristic: Maturity

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Age: «Time span over which a product has been developed»**.**	□	□	&	□
2. Continuity: «Regularity with which community contributions have been made to the a product or in relation to the product over its lifespan»	□	&	□	□
3. Activity on stable development branch: «Number and size of the contributions made to a product's stable development branch over a certain period of time»	□	&	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses maturity of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Age							4				
2. Continuity					6						
3. Activity on stable development branch					6						

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the maturity ?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses maturity of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Age							4				
2. Continuity					6						
3. Activity on stable development branch					6						

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the maturity ?

17.14 Characteristic: Community Robustness

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Maturity of security process: «Degree to which a development community has established processes dedicated to guarantee the security of delivered products».	□	&	□	□
2. Maturity of reliability process: «Degree to which a development community has established processes dedicated to guarantee that delivered products are free of critical defects (defects that prevent the operation of the product under common operation conditions)».	□	&	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the community robustness of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Maturity of security process					6						
2. Maturity of reliability process					6						

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the community robustness?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses the community robustness of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Maturity of security process					6						
2. Maturity of reliability process					6						

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the community robustness?

17.14.1 Characteristic: Maturity of the security process

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Compliance: «Degree to which the processes and procedures dealing with security adhere to best practices and security standards»**.**	&	□	□	□
2. Reaction time: «Amount of time that is typically required for resolving security-related issues »	□	&	□	□
3. Inclusion of preventive/reactive actions: «Degree to which the community commits to actions aimed at preventing security problems »	□	&	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses maturity of the security process of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Compliance		9									
2. Reaction time				7							
3. Inclusion of preventive/reactive actions				7							

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the maturity of the security process ?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses maturity of the security process of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Compliance		9									
2. Reaction time				7							
3. Inclusion of preventive/reactive actions				7							

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the maturity of the security process ?

17.14.2 Characteristic: Maturity of the reliability process

Question 1 : How meaningful are these definitions for you? Would you say completely meaningful, mostly meaningful, mostly meaningless or completely meaning less?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*
1. Compliance: «Degree to which the processes and procedures dealing with security adhere to best practices and security standards»**.**	&	□	□	□
2. Reaction time: «Amount of time that is typically required for resolving reliability-related issues »	□	&	□	□
3. Inclusion of preventive/reactive actions: «Degree to which the community commits to actions aimed at preventing reliability problems »	□	&	□	□

Question 2: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses reliability of the security process of a F/OSS component as an end-application?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Compliance				7							
2. Reaction time				7							
3. Inclusion of preventive/reactive actions				7							

Question 3: Considering a F/OSS component as an end-application , which other sub-characteristics do you consider relevant to asses the reliability of the security process ?

Question 4: Consider a scale from 0 to 10, where 10 represents completely relevant and 0 represent completely irrelevant, how relevant do you thing the following sub-characteristic are to asses reliability of the security process of a F/OSS component as a platform level?

Sub-characteristic	*Completely relevant.....Completely irrelevant*										
1. Compliance				7							
2. Reaction time				7							
3. Inclusion of preventive/reactive actions				7							

Question 5: Considering a F/OSS component as a platform level, which other sub-characteristics do you consider relevant to asses the reliability of the security process ?

17.15 Interview 6

17.16 Organizational Information

Date	Dec. 5th 2007
Start time	1:24 PM
Company	Organization E
Context / Domain	Research Centre
Interviewee Position	Research Scientist

1. Do you use F/OSS components as end-application: Yes

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

Embedded	External service	Internal service	Desktop	Development
.	.	X	X	X

2. Do you use F/OSS components as platform level (to build an application): Yes

   If your answer was YES, please marks the context in which the F/OSS is intended to be used:

Embedded	External service	Internal service	Desktop	Development
.	.	.	X	.

3. How do you use F/OSS components

   . Integrating an F/OSS product into a company's infrastructure

   X Integrating an OSS product/components into a software product/system developed by a company

   . Forking an existing open source component

   X Extending an open source product (to communicate with my product) to work in my environment

   X Selecting an open source language and libraries to develop my product on

18 Characteristic: Product Evolvability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
Product evolvability: "The ability of a product to be corrected, adapted and extended over time, according to the needs of its users".	X	□	□	□	
1. Usefulness of code documentation: «Extent to which the source code documentation (explicitly describing the product's internals) is useful when performing corrections, adaptations or extensions to the product».	□	□	X	□	See Comment 1. below
2. Usefulness of user documentation: «Extent to which the product's user/administrator oriented documentation is useful when deploying and using the product».	□	X	□	□	See Comment 2 below
3. Maintainability: «Amount of effort required by a programmer or team of programmers with no previous knowledge of the product, to understand its code to the point that successful modifications are possible».	□		X	□	See Comment 3 below
4. Portability: «Ease with which a system or component can be transferred from one hardware or software environment to another».	X	□	□	□	See Comment 4 below
5. Interoperability: «Degree to which a software product can interoperate with other software product either live or based on input/output data ».	□	X	□	□	See Comment 5 below
6. Compliance to standards: «Degree to which a product complies with published standards that are relevant to its functionality. Important note: for measurement purposes, this criterion is applied separately to various relevant software artefacts, i.e., source code, documentation, etc»	□	□	□	X	Does not match with our tree hierachy anymore Point 6 should be erased

Comment 0.

General Comment: We should limit our definition to a single style. That is currently we find definition that start with the ability of, extend to which, degree to which, ease with which.

Given that the goal of QualOSS is to assess and eventually give a score or a note to a characteristic, I believe the best formulation for starting our definition is "degree to which" (Furthermore it is also how SEI starts its definition so that can be bad)

My reason for preferring "the degree to which" is that it is a completely neutral way to state something where as "the ease with which" depends on the person (some people can easily do what others cannot + the term easy introduce ambiguity since what easy

means to one person is not the same to another, e.g., working 12 hours straight to solve a problem completely may be defined as hard by some one and easy by another)

Comment 1.

Furthermore, what do we mean by code documentation: just the doc in the code or do we also include other technical documentation: for example, what about technical documentation targeted to developers found in dev-forum or dev-mailinglists or even if rare, architecture description document.

What about test scripts?

Also, the term useful is hard to apprehend, we somewhat get the sense of what the definition is trying to achieve but if we can find another term or set of terms to substitute I believe the quality model would be improved.

Comment 2

The term "useful" is always ambiguous so if we could replace it by another term or even set of terms I believe the definition would gain in clarity and accuracy

Comment 3

First, why do we deviate from standard definition such as IEEE 610. (I would use IEEE 610 however to be consistent if we go with the formulation "degree to which", I would then replace the start from the definition from "the ease with which" to "degree to which".

Second, defining maintainability by "amount of work" is probably too constraining as that means our results should be able to estimate the amount of work needed to perform a particular maintenance. And as shown model such as COCOMO "amount of work" is influence by many factors other than the product, e.g., people's ability, people's knowledge of the domain and past experience with the product.

Comment 4

For consistency, I would start the definition with "the degree to which"

Comment 5

The definition deviates from IEEE 610 and ISO 9126 standard definition and I am not sure how they compare. IEEE defines interoperability as "The ability of two or more systems or components to exchange information and to use the information that has been exchanged". This seems to be at a higher level than ours. Again for consistency with our other definition here is how I would word it:

The degree to which two or more systems or components are capable of exchanging information and use the information that has been exchanged"

There is no need to differentiate between live and passive interoperability at this level. This can be done at the next level of our tree hierarchy.

What about a characteristic such as Compatibility how does it relate and how is it different from Interoperability

Question 2: When evaluating the product evolvability of OSS components, would you consider any additional characteristic?

No.	*Characteristic*	*Description*
	Community Capability and Experience	Experience of the community performing maintenance in general and in on the given F/OSS product in particular

	Evol-Centricity	The degree to which product evolution and maintenance is treated as central activities by the community, in the development process and reflected by the tool selected by the FlOSS endeavor (e.g. Have they performed refactoring or have planned to perform some refactoring, do they have procedure and recommendation regarding refactoring, do the recommend using IDE that have refactoring functions, do they recommend various reading to community members regarding refactoring, are there event where evolution and refactoring are presented or training is given by a community member to other members? ...

QUESTION 3 and 4 Skipped

18.1.1 Characteristic: Usefulness of code documentation

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Actuality: «Extent to which the code documentation describes the current version of the source code as opposite to describing older versions of it»**.**	☐	☐	☐	☐	Should Start with "the degree to which"
2. Coverage: «Ratio between size of documented code and general product code size»	☐	☐	☐	☐	Inadequate it is a metric not a definition. Here is a proposition: The degree to which code modules. units and elements are sufficiently documented (note that code includes build scripts and configuration)
3. Code documentation standard compliance: «Degree to which a product complies with published standards that are relevant to its functionality »	☐	X	☐	☐	I believe the last part of the definition "that ... its functionality" should be removed it make the definition confusing

Question 2: When evaluating the usefulness of code documentation of OSS components, would you consider any additonal characteristics?

No.	*Characteristic*	*Description*
4	Adequacy	The degree to which the semantic content of code documentation is adequate to gain understanding of the program
5	Possibly many other characteristics related to on other technical documentation e.g. Architectural description, design doc,.. (Found in developers forum, mailing list, etc.)	It is unclear whether code documentation realy means "code documentation" or really meant "technical documentation"
6	Test script documentation	We should consider test documentation, that is, documentation explaining what is being tested (especially useful for more complex tests probably not necessary for all unit tests)

18.1.2 Characteristic: Usefulness of user documentation

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Actuality: «Extent to which the user documentation describes the current version of the product functionality as opposite to describing outdated functionality»..	☐	☐	☐	☐	Replace "Extend" by "The degree"
2. Coverage: «Ratio between the number of documented product features and the general number of features offered by the product»	☐	☐	☐	☐	Inadequate it is specified as metric not plain definition The degree to which each feature of the product is sufficiently sufficiently documented
3. Internationalization: «Availability of the documentation in various natural languages»	☐	☐	☐	☐	
4. User documentation standard compliance: «Degree to which a product complies with published standards relevant to documentation»	☐	☐	☐	☐	

Question 2: When evaluating usefulness of user documentation of OSS components, would you consider any additional characteristic?

No.	*Characteristic*	*Description*
	Doc Understandability	The degree to which the end-user documentation is adequately written for its target audience
	Organization	The degree to which the documentation is partitioned and organized to facilitate the findings of information for the various type of users (or administrators)

18.1.3 Characteristic: Maintainability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Product complexity: «The Degree to which a system or component has a design or implementation that is difficult to understand and verify»	□	□	□	□	
2. Architecture flexibility: «Ability of the product's architecture of being applied to new problems. The ease which with a system or component can be modified for use in applications or environments other than those for which it was specifically designed»	□	□	□	□	Start with the degree to which
3. Fix ability: «Ease with which a software product can be fixed»	□	□	□	□	How is it different from Changeability?
4. Maintainability standard compliance: «Degree to which a product complies with published standards relevant to maintainability»	□	□	□	□	
5. Product buildability: «Degree to which a system or component can be rebuild after modifications to the source»	□	□	□	□	

Question 2: When evaluating the maintainability of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*
	General Comment	Product Complexity is not a sub-characteristic of maintainability, it is a factor or an attribute that influence maintainability, for sub-characteristics, we should look at ISO9126. Nonetheless, I believe that product complexity as well as architecture flexibility are important factor to ask questions anout and to showhow measure .
	Analysability	*Attributes of software that bear on the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.* (ISO 9126: 1991, A.2.5.1) The main problem with the ISO definition is that it remains fuzzy since one

		still has to determine the "Attributes". In turn, this mean we can probably point to the fact that analysability is most likely influenced by the attribute product complexity, architecture modularity, etc.
	Changeability	*Attributes of software that bear on the effort needed for modification, fault removal or for environmental change.* (ISO 9126: 1991, A.2.5.2) Similar to Analysability, we must determine the attributes that likely influence changeability, for example, product complexity, architecture flexibility, etc.
	Effectibility	I replace Stability with Effectibility I believe Stability has a different connotation to most people. It is more often used to refer to the ability of software to stay up and running (hence closer in meaning to robustness than maintainability). Instead of stability, the term "Effectability" could be used and the replacement definition could be "the degree to which the effect of a modification can be completely anticipated" this is somewhat equivalent.
	Testability	*Attributes of software that bear on the effort needed for validating the modified software.* (ISO 9126: 1991, A.2.5.4)

Characteristic: Portability

Question 1 : How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Platform specificity: «Degree to which a product's code is specific to a particular hardware or software environment»	□	□	□	□	
2. Standard compliance: «Degree to which a product complies with published standards that are relevant to its functionality»	□	□	□	□	Need to replace the word "functionality" by something else, "purpose" maybe

Question 2: When evaluating the portability of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*
	General Comment	Our list of sub-characteristics is not the same as that of ISO9126
	Adaptability	*Attributes of software that bear on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software considered.* (ISO 9126: 1991, A.2.6.1)
	Installability	*Attributes of software that bear on the effort needed to install the software in a specified environment.* (ISO 9126: 1991, A.2.6.2)

	Replaceability	*Attributes of software that bear on the opportunity and effort of using it in the place of specified other software in the environment of that software.* (ISO 9126: 1991, A.2.6.4)
		For all 3 sub-charactersitics, we must sill identified the “Attributes” but they needn't be yet another layer of characteristics rather, this is where we should generate questions (from the GQM). So for example, we first identify the attributes that influence Adaptability and then we can create a set of questions related to each attribute that will help measure it.

Characteristic: Interoperability

Question 1: How meaningful are these definitions for you? / Do the following definitions make sense to you? If a characteristic isn't ranked as completely meaningful, ask WHY? What you would change or add?

Sub-characteristic	*Completely meaningful*	*Mostly meaningful*	*Mostly meaningless*	*Completely meaningless*	*Comments*
1. Runtime interoperability: «Interoperability with other software products while in operation»	□	□	□	□	The definition would be more accurate as follows. The degree to which a software system is capable of exchanging information with other systems and using it while all systems are in operation
2. Passive interoperability: «Interoperability with other software products based on output data generated by the software product or based on the capacity of the software product to read various data types and formats»	□	□	□	□	The degree to which a software system is capable of using information generated by other systems in an asynchronous mode i.e., when systems are not running at the same time.

Question 2: When evaluating the interoperability of OSS components, do you consider any additional characteristic?

No.	*Characteristic*	*Description*