


 <p>Sponsored through Framework Programme Sixth (Call 5) by</p> <div style="display: flex; justify-content: space-around; align-items: center;">   </div>		<b>Document Information</b>
		<b>Date</b> : Mar 15, 07 <b>Version:</b> 1.3 <b>Pages</b> : 88 <b>Revision:</b> 292
		<b>Owning Partner:</b> IESE
		<b>Author(s):</b> Marcus Ciolkowski, Martín Soto (IESE), Jean-Christophe Deprez
		<b>Reviewer(s):</b> FUNDP
		<b>To:</b> CONSORTIUM
		<b>Purpose of distribution:</b>
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)		<b>Printed</b> <b>on 03/15/07 at 05:30:24 PM</b>
<b>Status:</b> <input type="checkbox"/> Draft <input type="checkbox"/> To be reviewed <input checked="" type="checkbox"/> Proposal <input type="checkbox"/> Final/Released	<b>Confidentiality:</b> <input checked="" type="checkbox"/> Public - Intended for public use <input type="checkbox"/> Restricted - Intended for QUALOSS consortium only <input type="checkbox"/> Confidential - Intended for individual partner only	
<b>Deliverable ID:</b> <b>D1.2</b>  <b>Title:</b>  <p style="text-align: center;">Measurement Requirements Specifications          (Specification of Goals for the QualOSS Quality Model)</p> <p style="text-align: center;">A deliverable of Task 1.2</p>		

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 2 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

## **Deliverable: D1.2**

### **Title: Measurement Requirements Specifications**

#### **Executive Summary:**

This document describes the work done and results obtained in task 1.2 ("Goal of measurements") of the QualOSS project. We review relevant definitions of robustness and evolvability in F/OSS assessment approaches and in the state of the art and practice of quality models. Additionally, we take into account stakeholders' perceptions and requirements through a series of interviews.

Goals and requirements for the QualOSS model are defined in terms of (a) business and measurement goals, and (b) a consolidated definition of quality characteristics for evolvability and robustness from related work and from stakeholders' views, and (c) an initial plan for validation of the QualOSS model.

Further work is still required. In particular, the QualOSS model needs to be further refined into metrics; this is part of task 1.3. We foresee that part of task 1.3 will be to develop an assessment method for evaluating the community maturity, as approaches to evaluate associated processes have so far not been considered in F/OSS assessment methods.

Section 1 presents the motivation of task 1.3, and explains how the tasks in workpackage 1 collaborate to produce the initial QualOSS model.

Section 2 defines the terminology used in the remainder of the report.


Section 3 presents related work on quality models relevant for defining robustness and evolvability of F/OSS project. It is divided into two parts: One part is focused on assessment methods for F/OSS projects (QSOS; OSM, and OpenBRR), while the second part describes the state of the art and practice in quality modelling (McCall and Boehm, DGQ, FURPS, NASA SATC, and ISO 9126).

Section 4 describes robustness and evolvability from stakeholders' viewpoints. In particular, it presents the results of a series of interviews conducted as part of task 1.2. These interviews were aimed at deriving usage scenarios for F/OSS components (i.e., how stakeholders intended to use F/OSS products), and at evaluation criteria used by stakeholders (i.e., at how stakeholders evaluate whether F/OSS products are suitable).

Section 5 presents the consolidated requirements for the QualOSS model; including the initial version of business and measurement goals, and the resulting QualOSS definition of robustness and evolvability that will be used as basis for D 1.3.

Section 6 contains the initial validation plan, while Section 7 summarizes and concludes this report.

The Appendix contains the questionnaire used to conduct the interviews, as well as the interview results.


	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 3 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

## CHANGE LOG

Ver.	Date	Author	Description
0.1	15/09/06	Marcus Ciolkowski	Initial proposal for structure
0.2	20/09/06	Jean-Christophe Deprez	Add notes to Section 3
0.9	08/03/07	Marcus Ciolkowski, Martín Soto	Draft version
1.0	12/03/07	Flora Kamseu	First review
1.1	13/03/07	Marcus Ciolkowski	Rework and completion
1.2	15/03/07	Flora Kamseu	Second review
1.3	15/03/07	Marcus Ciolkowski	Rework, anonymization, security aspects refined


## APPLICABLE DOCUMENT LIST

Ref.	Title, author, source, date, status	Deliverable Identification


	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 4 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

## TABLE OF CONTENTS

<b>1 Introduction.....</b>	<b>6</b>
1.1 Motivation.....	6
1.2 Goal.....	6
1.3 Strategy For Workpackage 1.....	6
1.4 Approach.....	8
1.5 Structure of the Deliverable.....	8
<b>2 Terminology.....</b>	<b>9</b>
2.1 GQM.....	9
2.2 Software Quality.....	12
2.3 Software Quality Models.....	13
2.4 Structure of a Quality Model.....	15
2.5 Measurement.....	17
<b>3 Quality Models State of the Art.....</b>	<b>17</b>
3.1 Definition of Evolvability and Robustness from F/OSS Assessment Methodologies.....	17
3.1.1 QSOS .....	17
3.1.2 Open Source Maturity Model (Cap Gemini) .....	20
3.1.3 Open Business Readiness Rating (OpenBRR) .....	21
3.1.4 Comparing QSOS, OSMM and OpenBRR.....	24
3.2 Definition of Evolvability and Robustness from the State of the Art and Practice.....	28
3.2.1 The Historic Models: McCall and Boehm.....	28
3.2.2 Quality Model of the Deutsche Gesellschaft für Qualität.....	30
3.2.3 FURPS/FURPS+.....	33
3.2.4 Quality Model of the NASA SATC.....	37
3.2.5 The Standard: ISO9126.....	38
3.3 Discussion.....	39
<b>4 Results From Interviews.....</b>	<b>39</b>
4.1 Usage Scenarios .....	40
4.2 Quality Model From Interviews.....	41
<b>5 Consolidated Measurement Requirements for QualOSS .....</b>	<b>42</b>
5.1 Business Goals.....	42
5.2 Measurement Goals.....	43
5.3 Definition of QualOSS Quality Indicators.....	44
5.3.1 Rationale.....	45
5.3.2 Evolvability.....	45
5.3.3 Robustness.....	48
5.3.4 Quality Factors Not Considered by QualOSS.....	50
<b>6 Validation Plan.....</b>	<b>50</b>
6.1 Goal Category 1: Evaluation of appropriateness of QualOSS model.....	50
6.2 Goal Category 2: Evaluation of QualOSS model impact:.....	51
<b>7 Summary and Conclusions.....</b>	<b>52</b>
<b>References.....</b>	<b>54</b>
<b>Appendix A: Usage Scenario Interview Sheet.....</b>	<b>57</b>
Instructions.....	57

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 5 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

Organizational Information.....	57
Interview Introduction.....	57
Interview Questions.....	57
<b>Appendix B: Interview Results.....</b>	<b>61</b>
Interview 1.....	61
Organizational Information.....	61
Interview Questions.....	61
Interview 2.....	64
Organizational Information.....	64
Interview Questions.....	64
Interview 3.....	65
Organizational Information .....	65
Interview Questions .....	65
Interview 4.....	68
Organizational Information .....	68
Interview Questions .....	68
Interview 5.....	70
Organizational Information .....	70
Interview Questions .....	71
Interview 6.....	73
Organizational Information.....	73
Interview Questions.....	74
Interview 7.....	75
Organizational Information.....	75
Interview Questions.....	76
Interview 8.....	78
Organizational Information.....	78
Interview Questions.....	78
Interview 9.....	83
Organizational Information.....	83
Interview Questions.....	83
Interview 10.....	85
Organizational Information.....	85
Interview Questions.....	85

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 6 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

## 1 INTRODUCTION

### 1.1 MOTIVATION

The strategic objective of the QualLOSS 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, QualLOSS notes that many organizations integrate Free *libre* Open Source Software (F/OSS) in their systems hence QualLOSS aims at facilitating the selection of the most adequate F/OSS . In particular, QualLOSS focuses on assessing the evolvability and robustness of F/OSS.

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 QualLOSS 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, D1.2, is to define the goals and requirements for this assessment method, the QUALOSS quality model.

### 1.2 GOAL

The goal of task 1.2 in workpackage 1 is to determine the goals companies want to achieve when measuring evolvability and robustness of software components with the objective to integrate these components in their software systems and products. Part of these goals are business goals related to F/OSS usage. Another part of these goals is the stakeholders' definition of robustness and evolvability; that is, what evaluation criteria they use.


The key result of task 1.2 is a definition of robustness and evolvability based on related work and stakeholders' views. These definitions will be represented in the form of quality characteristics relevant to evolvability and robustness. Definition of concrete metrics for these quality characteristics is part of task 1.3.

### 1.3 STRATEGY FOR WORKPACKAGE 1

The main objective of WP1 is to perform requirement analysis through prototyping. Currently, there exists a set of metrics and corresponding measuring tools.

The outcome of prototyping in WP1 serves in performing a thorough requirement analysis in order to well formulate our requirements and eventually, 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 models constructed on basic metrics and the calibration exercise yield interesting results directly usable and transferable to our QUALOSS platform then that is 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.

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 7 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

(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 models is addressed in task 1.3. The definition will be 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 on available models. 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 will also take into account available data and tools, as elicited in task 1.1. Figure 1 shows the inputs for task 1.3. In particular, this implies that, compared to the description of work, the definition of metrics for the QualOSS model will completely be shifted to task 1.3.

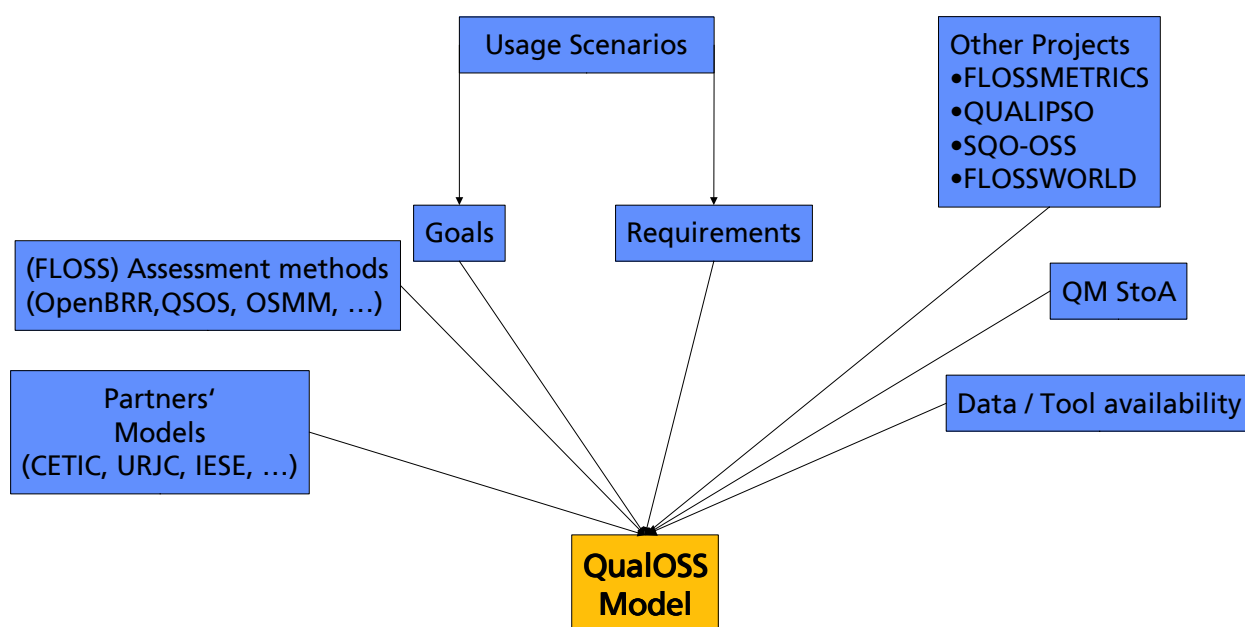



Figure 1: Input Sources for QualOSS model (D 1.3)

(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. Calibration of the quality models is addressed in task 1.5. More precisely, task 1.5 examines the usefulness and applicability of the quality models and tries to find patterns and dependencies in the data that can be used as input to improve the quality models. Task 1.6 validates the quality models on additional projects. This includes, for example, evaluating the definition and prioritization of quality characteristics from stakeholders' viewpoints. Workpackages 4 and 5 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. For example, stability of a product is measured in the same way for all products; however, if it

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 8 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

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. Figure 2 illustrates the dependency between D 1.2 and D 1.3.

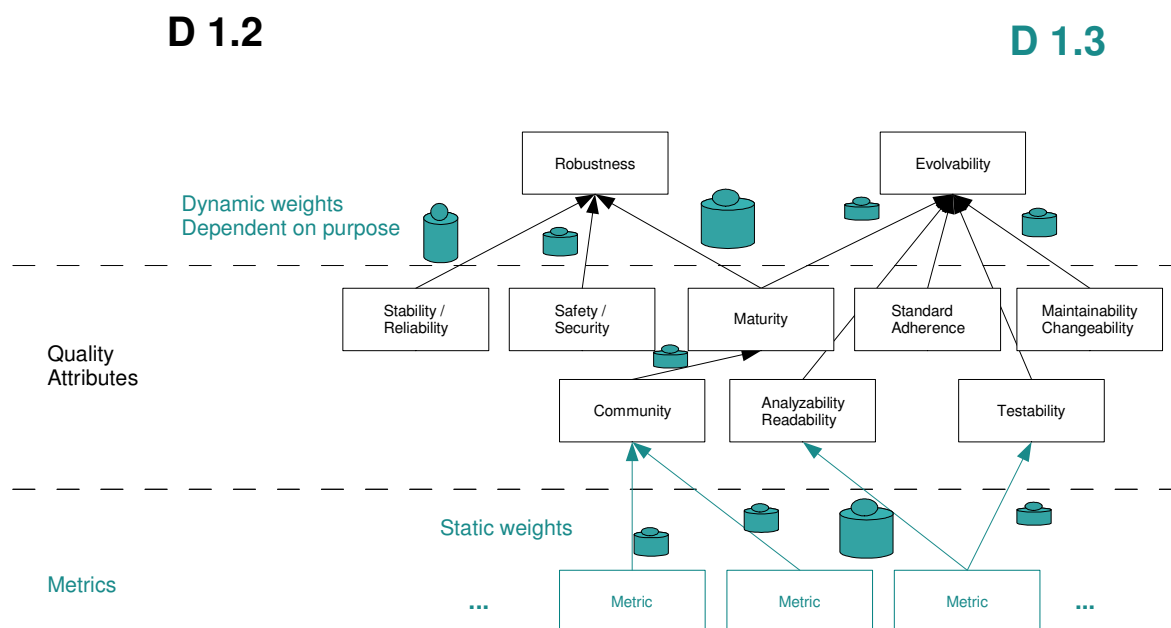


Figure 2: Relation between Deliverables 1.2 and 1.3: D 1.2 defines the quality characteristics that are relevant for evolvability and robustness, while D 1.3 contributes the definition of metrics and an initial proposal of weighting schemes to reflect different priorities between quality characteristics

## 1.4 APPROACH

This section describes the approach we took to achieve the goals of Deliverable 1.2.


The goals of D1.2 can be summarized as follows: Determine the goals companies want to achieve when measuring evolvability and robustness of software components with the objective to integrate these components in their software systems and products.

The approach taken in task 1.2 is to first define and elicit usage scenarios for OSS components from stakeholders through structured interviews. As a next step, we define evolvability and robustness based on these scenarios as well as on related work in quality modelling and assessment methods for F/OSS projects. The result of this work is a definition of QualOSS quality indicators for evolvability and robustness in terms of quality characteristics and sub-characteristics that correspond to GQM measurement goals and GQM questions.

Finally, task 1.2 defines an initial validation plan that will be further refined in tasks 1.4-1.6. Workpackages 4 and 5 will build upon and further refine the validation plan proposed here.

## 1.5 STRUCTURE OF THE DELIVERABLE

The rest of the deliverable is structured as follows:

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 9 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

Section 2 defines the terminology used in the remainder of the report. Section 3 presents related work on quality models relevant for defining robustness and evolvability of F/OSS project. It is divided into two parts: One part is focused on assessment methods for F/OSS projects (QSOS; OSM, and OpenBRR), while the second part describes the state of the art and practice in quality modelling (McCall and Boehm, DGQ, FURPS, NASA SATC, and ISO 9126). Section 4 describes robustness and evolvability from stakeholders' viewpoints. In particular, it presents the results of a series of interviews conducted as part of task 1.2. These interviews were aimed at deriving usage scenarios for F/OSS components (i.e., how stakeholders intended to use F/OSS products), and at evaluation criteria used by stakeholders (i.e., at how stakeholders evaluate whether F/OSS products are suitable). Section 5 presents the consolidated requirements for the QualOSS model; including the resulting QualOSS definition of robustness and evolvability that will be used as basis for D 1.3. Section 6 contains the initial validation plan, while Section 7 summarizes and concludes this report. The Appendix contains the questionnaire used to conduct the interviews, as well as the interview results.

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

## 2 TERMINOLOGY

This chapter introduces a consistent set of terms related to quality modelling. The purpose of this introduction is twofold. First, it defines a framework that structures the contents of the subsequent chapters of this report, and, second, it aims at consolidating the terminology presented in the literature, which is often used inconsistently across different sources.

Section 2.1 gives an overview of the GQM approach. In Section 2.2 and 2.3, quality terms are defined. In Section 2.4 a framework for quality models is presented. In Section 2.5 measurement related terms are defined.


### 2.1 GQM

The Goal Question Metric (GQM) paradigm (Basili et al., 1994; Basili and Rombach, 1988; Basili and Weiss, 1984) has been proposed as a goal-oriented approach for measuring product and process qualities in software engineering. It is a top-down mechanism for defining a set of goals and for evaluating them using measurement. GQM represents a systematic approach for adapting and integrating goals with models of the software processes, products and quality perspectives of interest, based upon the specific needs of the project and the organization.

The GQM paradigm was first developed at the University of Maryland in 1984 (Basili and Weiss, 1984) in cooperation with the NASA Goddard Space Flight Center (Basili et al., 2002) and has been extended as part of the TAME project (Basili and Rombach, 1988). It is intended as the goal setting step in an evolutionary quality improvement paradigm tailored for a software development organization, the Quality Improvement Paradigm (QIP) (Basili et al., 1994).

A first formal description of how to apply GQM was defined by Basili in 1992 (Basili, 1992) and was later refined (van Solingen, 1999, Gresse et al., 1995). A GQM quality model has a hierarchical structure. It consists of three components: goal, question, and metric (see Figure 3). Each of those corresponds to a different level (conceptual, operational, and quantitative).

**Conceptual level (Goal):** A goal is defined for an *object*, for a variety of *purposes*, with respect to various *models of quality*, from various *points of view*, relative to a particular *context*. The GQM-Goal-Template is presented in Table 1.

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 10 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

<b>GQM- Goal-Template</b>	
<b>Object</b>	Products, Processes, Resources
<b>Purpose</b>	Characterization, Improvement, Prediction, ...
<b>Quality focus</b>	Reliability, Costs, Usability, Maintainability, ...
<b>Point of view</b>	Project manager, Developer, User, Upper management
<b>Context</b>	Project X of company A

Table 1: GQM Goal Template

**Operational level (Questions):** A set of questions is formulated to break down the goal. Various aspects of the quality focus are characterized and factors that might influence the quality (i.e., influencing factors) are identified. The questions support the evaluation and interpretation of the measurement data regarding the goal.

**Quantitative level (Metric):** A set of data is associated with every question in order to answer it in a quantitative way.

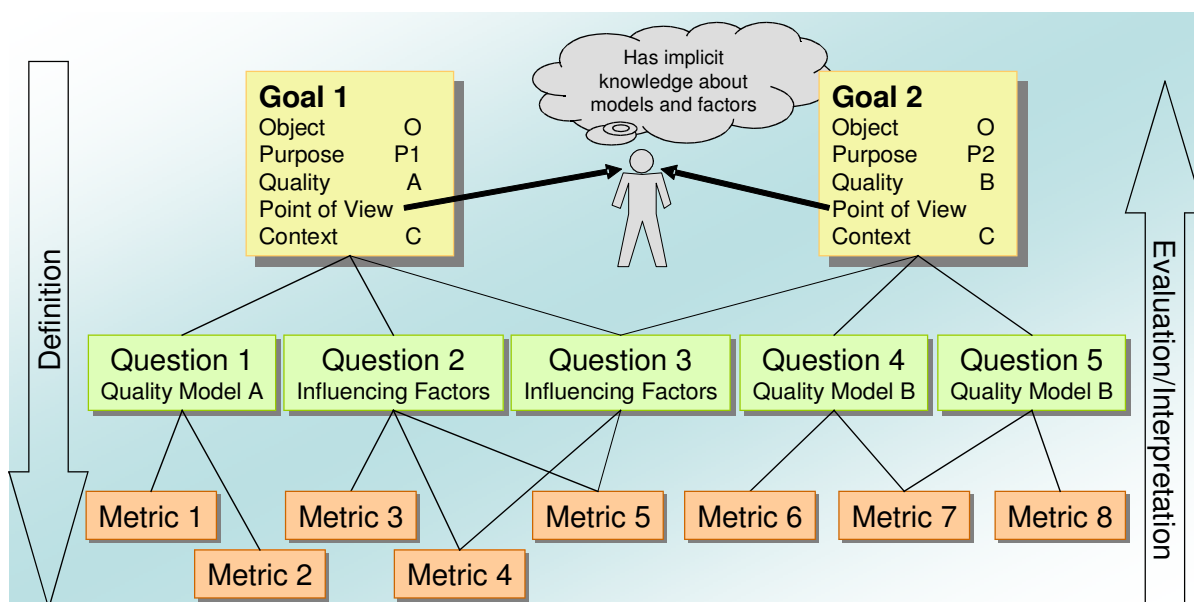



Figure 3: GQM Quality Model. The yellow color represents goals (conceptual level), green represents questions (operational level), and red represents metrics (quantitative level).

In order to systematically apply the GQM-Quality Model in an organizational and project setting, a GQM-Process was defined (Basili, 1992, Gresse et al., 1995). According to (Gresse et al., 1995), the GQM-Process is decomposed to the following steps:

**Perform Prestudy:** The objective of this step is the collection of information which is relevant in the context of quality modelling for the organization and its projects. For example, the organizational and project goals must be identified, (e.g., what goal is more important for the organization: keeping the delivery date or meeting quality demands).

**Identify GQM-Goal:** Based on the organizational and project goals different goals are formulated according to the GQM-Goal Templates and ranked in a priority list. In the pilot phase of quality modelling only one maximally two should be selected.

**Produce GQM quality model:** The implicit knowledge with respect to the goal is elicited by interviewing the people stated as point of view in the goal template. The information resulting from all interviews is merged. This merged information is used to refine the GQM quality model. Existing conflicts have to be clarified.

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 11 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

Inconsistencies and missing items are resolved by interviewing people again. Abstraction sheets are a means for collecting and storing the information (see Gresse et al., 1995).

**Produce Measurement Plan:** A measurement plan states a concise specification of the data collection process that encompasses who collects at what point in time which kind of data with which aids. The main issue that is to be addressed is the appropriate integration of measurement into the processes performed in the software project.

**Collect and Validate Data:** The measurement data are collected according to the defined procedures in the measurement plan. The collected data is validated and stored, best in electronic form.

**Analyze Data:** The analysis and interpretation of the collected data is conducted during feedback sessions (i.e., a meeting of the people representing the point of view in the goal template and the data providers). The data, raw or graphically processes, is studied in advance by the participants of the feedback sessions. During the feedback sessions the presented material is analyzed and interpreted by the people in the point of view of the goal template. The results are conclusions with respect to the goal under investigation.

**Package:** The results and experiences gained with respect to the goal under investigation are fixed to make them available in upcoming projects. The context is stated in which the results and experiences are valid and/or can be applied.


In the application of the GQM approach, the permanent interaction between project team and the measurement responsible is crucial as well as the placement of measurement rationales and results at other or upcoming project's disposal. To support the communication between project and measurement staff, a representation called abstraction sheet was developed. As side effect and to avoid a costly build up of a measurement database, the abstraction sheet can be used as storage medium to ensure a long-lasting benefit across projects. An abstraction sheet summarizes the main issues and dependencies of a GQM quality model and presents this information on four quadrants of a sheet (see Figure 4):

**Aspects of Quality focus and Metrics:** What does the quality focus specified in the GQM goal template mean to the project representatives? What are metrics to analyze the quality focus?

**Baseline hypothesis:** What is the estimate on the outcome of the quality aspect?

**Variation factor** What are influencing factors that have an impact on the quality focus?

**Impact on baseline hypothesis** What is the (assumed) impact of an influencing factor on the quality focus?

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 12 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

Goal	Object	Purpose	Quality focus	Point of view	Context
	SW-Product alpha	Understanding	Reliability	SW development team	Company X Project delta
<b>Aspects of quality and metrics:</b> 1. Number of failure - total number - severity (sorted by criticality) 2. Number of faulty modules - total number - sorted by modules - sorted by lifecycle phases of detection			<b>Influencing factors:</b> 1. Degree of Reuse 2. Experience of Development Team Members 3. Adherence to Code Inspection Process		
<b>Baseline hypothesis:</b> 1. Number of failure - total number: 115 - estimates John: 30% critical, 70% uncritical - estimates Elsa: 15% critical, 60% uncritical, 15% others 2. Number of faulty modules - total number: 200 Modules - AlphaH (40 faults), AlphaD (25), AlphaF (10), AlphaX (9), .... - 10 in requirements phase, 30 in design, 50 implementation, 110 testing/integration			<b>Impact of influencing factors on baseline hypothesis</b> 1. The higher the degree of reuse, 2. The longer the experience of the development team members, the lower the number of failures and faults. 3. The closer the adherence to the code inspection process, the lower the number of failures and faults in testing and integration		


Figure 4: Abstraction Sheet

## 2.2 SOFTWARE QUALITY

According to the ISO/IEC standard 8402 (ISO 8402), **quality** is defined in general as “*the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs*”. Since this report is concerned with the quality of software products, it is also appropriate to introduce a specific definition for software quality. According to the ISO/IEC standard 9126 (ISO 9126), **software quality** is “*the totality of characteristics of a software product that bear on its ability to satisfy stated and implied needs.*” In the remainder of this report we will restrict ourselves to software quality.

A quality model has to take into account the distinction between internal and external quality. **External quality** is defined by the ISO/IEC standard 14598 (ISO 14598) as “*the extent to which a product satisfies stated and implied needs when used under specified conditions*”, that is, external quality represents the *user's view* on the system. It follows from the definition, that external quality can only be determined when the product already exists. On the other hand, the possibility of determining quality during the earlier phases of a product's development is also very desirable from a *developer's point of view*. This is because it is valuable for the developer to (1) get an early impression of software quality (before the system is delivered to the customer) and (2) determine and evaluate how development practices impact product quality. Therefore, ISO 14598 defines **internal quality** as “*the totality of characteristics of a product that determine its ability to satisfy stated and implied needs when used under specified conditions*”. Internal quality is important to get an early impression of quality and to make inferences on external quality.

In addition to internal and external quality, which refer to properties of a product itself, there is also the notion of quality-in-use, which corresponds to the users' view of the software when it is operating in the usage environment. Thus, quality-in-use is based on the actual *usage* of the software, rather than on intrinsic properties of the software itself. ISO 9126 defines **quality in use** as “*the capability of the software product to enable specified users to achieve specified goals with effectiveness, productivity, safety, and satisfaction in specified contexts of use*”. In the remainder of this report quality in use will not be discussed in detail.

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 13 of 88</p>
		<p>Version: 1.3 Date: Mar 15, 07</p>
		<p>Status : Proposal Confid : Restricted</p>

The relationship between software internal and external quality is illustrated by Figure 5. The figure stresses the fact that internal quality is an important indicator for external quality.

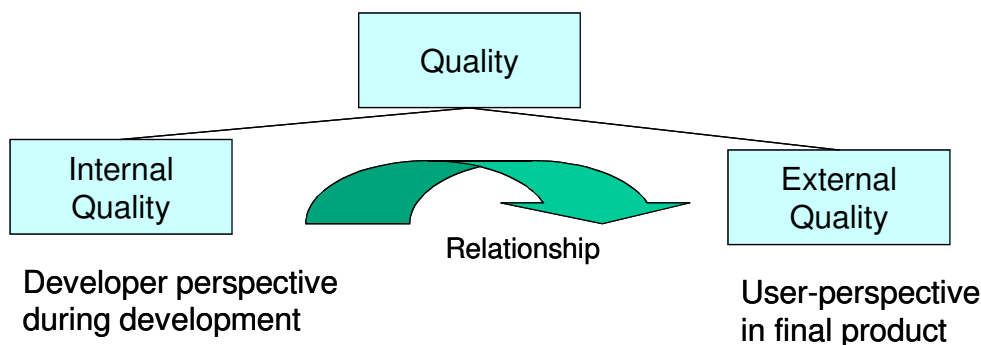


Figure 5: Internal and External Quality


The definitions taken from ISO 9126 and ISO 8402 are very general. For practical use a more detailed definition of quality is necessary. This is due to two main reasons:

- Quality has many facets and comprises many characteristics. It is necessary to define which of those characteristics contribute to software quality.
- In order to specify quality requirements or to assess a system's quality, it is necessary to define levels of quality and to verify their fulfilment. This calls for an operational, measurable definition of quality characteristics.

This leads us to the concept of a software quality model, whose main purpose is to operationalize the general definition of quality.

## 2.3 SOFTWARE QUALITY MODELS

According to ISO14598, the term **quality model** is defined as *“the set of characteristics and relationships between them, which provides the basis for specifying quality requirements and evaluating quality”*. Due to the generality of this definition, a variety of concepts can be potentially identified as quality models, a fact that often results in misunderstandings. This section's objective is to provide a comprehensive, yet clear definition of the various aspects of a quality model.

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 14 of 88</p> <hr/> <p>Version: 1.3 Date: Mar 15, 07</p> <hr/> <p>Status : Proposal Confid : Restricted</p>
---	--	--

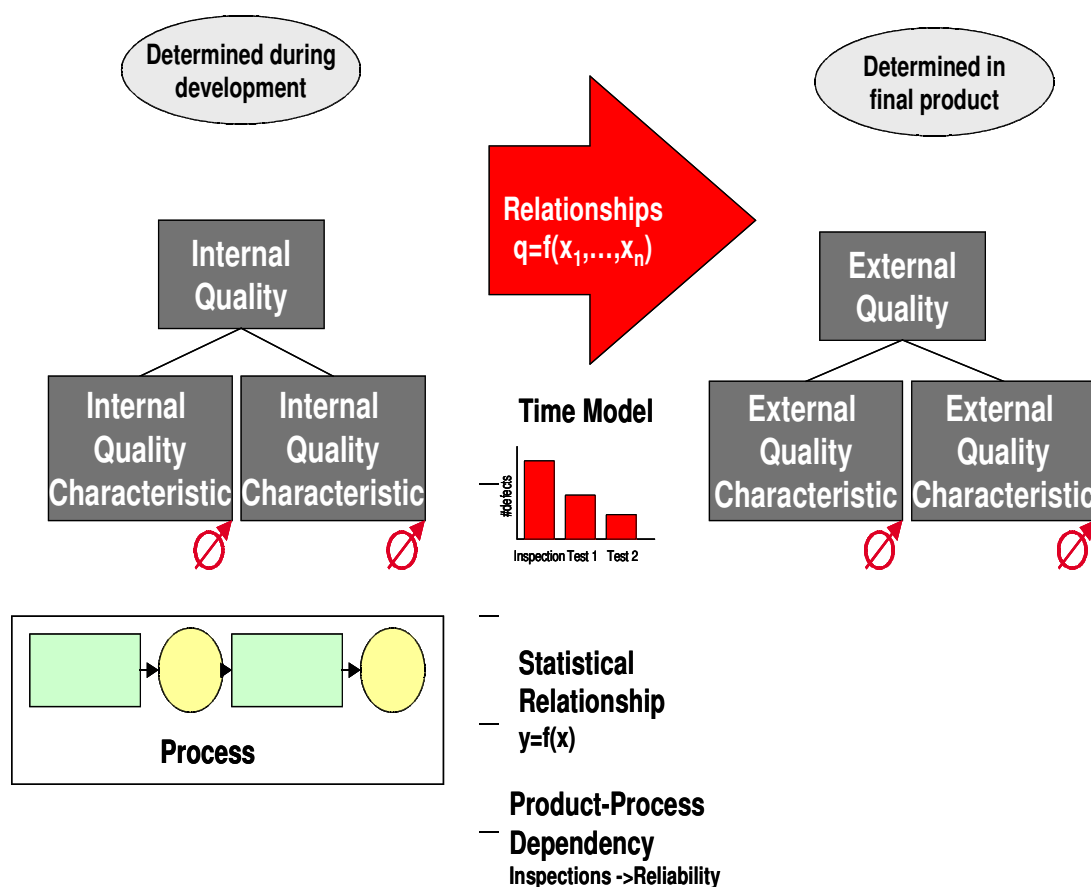


Figure 6: Software Quality Model Framework

Figure 6 depicts the components of a comprehensive quality model. The set of characteristics that constitute quality is captured by means of **product quality models**. Product quality models determine how the term quality is decomposed into quality characteristics. In order to be operational, these quality characteristics need to be measurable. Product quality models can be defined for internal and external quality.


**Quality relationships** can be seen as high-level functions  $q=f(x_1, \dots, x_n)$  that relate quality to its impacting factors. Such impacting factors, denoted as  $x_i$  in the figure above, can be measured earlier in the development process (e.g., internal quality) or even be part of the development process itself. The quality relationships can range from more qualitative relationships (e.g., knowing how a process impacts quality) to quantitative relationships (e.g., statistical prediction models). Typically, empirical studies such as case studies and experiments are necessary in order to identify and validate such relationships.

Several types of relationships are depicted in Figure 6.

**Time models** describe the expected behavior of a certain software measure (related to a quality characteristic) as a function of time. For example, a table listing the proportion or number of defects in each development phase can constitute a defect detection (time) model.

**Functional relationships** (also referred to as **statistical prediction models**) describe the relationship between various software qualities by means of an equation. These relationships provide a method for predicting the values of software qualities based on information that is more readily available or known. An example of a functional relationship is a fault-proneness model  $fp=f(\text{coupling, cohesion})$  that tries to predict the number of defects in a component depending on certain levels of coupling and cohesion.

**Product-process dependencies** are validated dependencies that describe how practices in the development process impact product quality. An example of such a product-process dependency can be

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 15 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

formulated as: “The development practice *Software Inspections* performed in the process *Software Requirements Analysis* impacts the product quality *Reliability*”. It has to be noted that product-process dependencies can only be determined from an actual process in a concrete environment. Using product-process dependencies for prediction in a new project is only possible if it exhibits only little variation with respect to the older project where the dependency was valid. Consequently, a standardized development process is a prerequisite for this kind of quality relationships to be useful in practice.

## 2.4 STRUCTURE OF A QUALITY MODEL

Quality models consist basically of quality characteristics, which are refined into quality sub-characteristics, and finally into measurable properties. Figure 7 Shows the structure of a quality model, as defined by the SQUID project. The SQUID quality model, which was defined in the late 1990s in an ESPRIT research project, has two components:

- A structure model that defines model elements and their interactions.
- A content model that identifies a set of entities linked in accordance with that structure.

The quality model refines the quality requirements defined for a certain product. This is done by defining suitable quality characteristics which are decomposed into sub-characteristics until they are directly measurable (also called quality attributes or properties). Quality (sub-)characteristics and attributes can be internal and external. Thereby, internal quality is defined as “the totality of characteristics of the software product from an internal view”. In contrast to internal quality, external quality is defined as “the totality of characteristics of the software product from an external view. It is the quality when the software is executed, which is typically measured and evaluated while testing in a simulated environment with simulated data using external metrics” (see Section 3.2.5 on ISO 9126). The user identifies how internal characteristics/attributes influence external ones by linking them in the model.

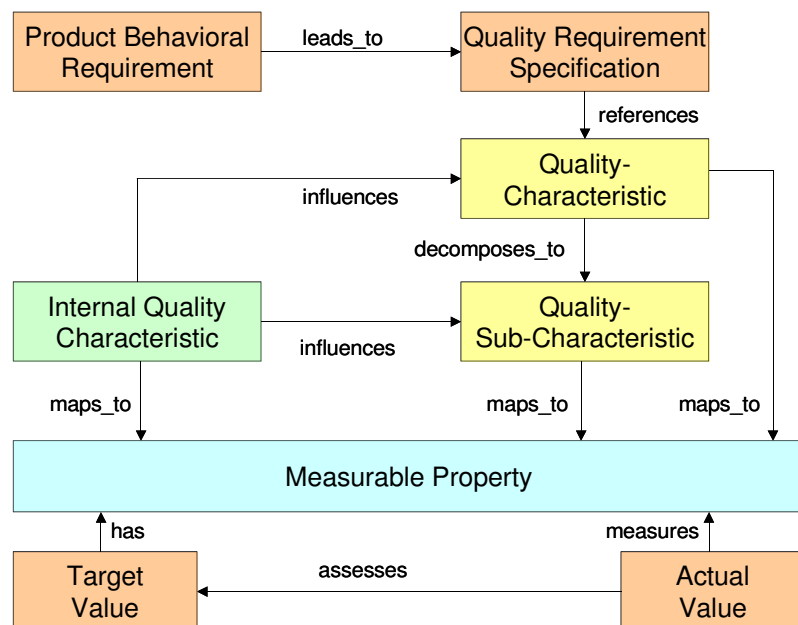



Figure 7: SQUID structure model of a quality model

The heart of SQUID is a general data model which defines what kind of information regarding the product and its quality should be identified and measured. Therefore, the data model consists of a context model and a quality model (see Figure 8). This division is proposed because different products or even different parts of the same product usually have different behavioural requirements and thus different quality requirements. The instantiation of this data model for a specific project is then called product quality model.

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 16 of 88</p> <hr/> <p>Version: 1.3 Date: Mar 15, 07</p> <hr/> <p>Status : Proposal Confid : Restricted</p>
---	--	--

The context model includes the deliverables (such as specifications and code) that are produced during the life of the project, the activities producing the deliverables, and the review points that are control interval. Quality measurement takes place of the deliverables at the specified review points. Furthermore, the quality requirement for the particular product is stated together with the target, actual, and estimated value. Target and estimated value might be provided by the experience base from similar products.

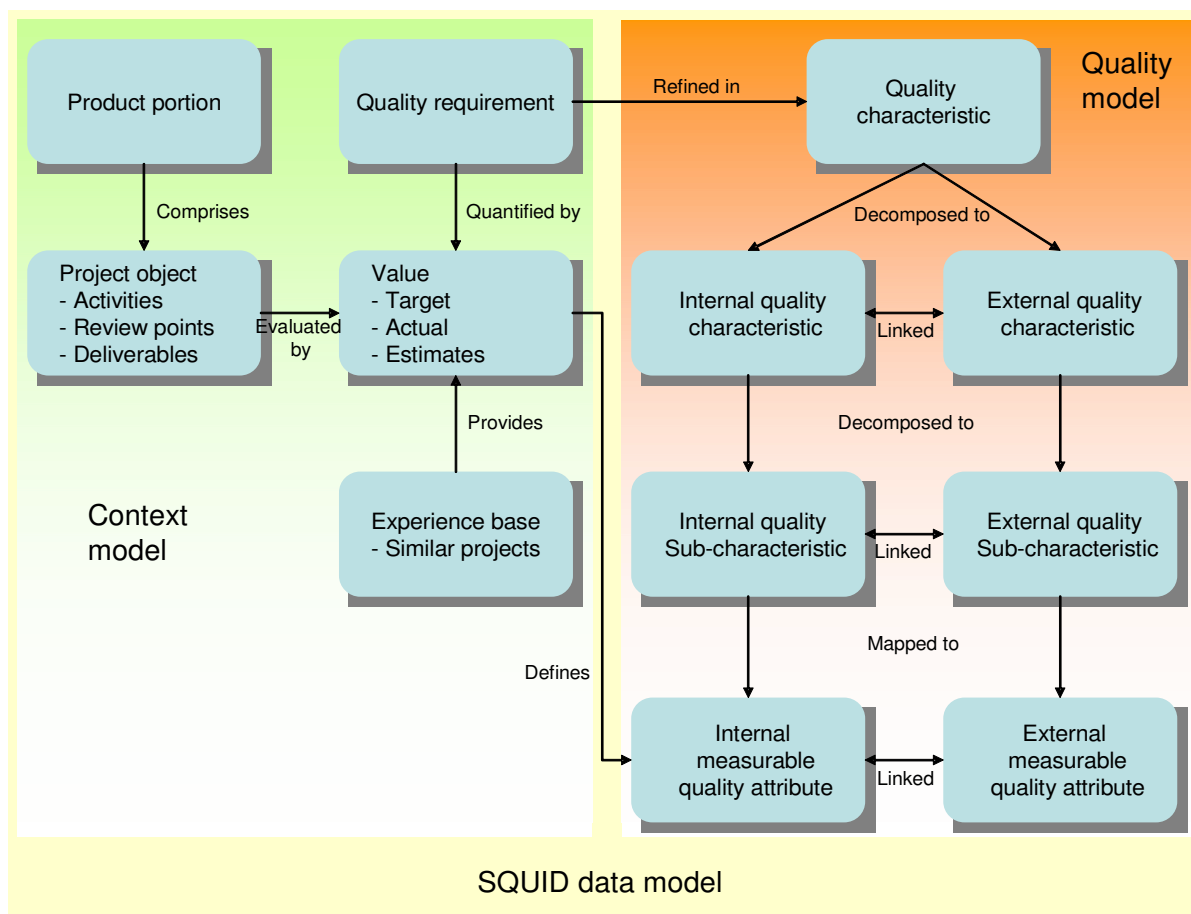



Figure 8: SQUID context model

An important, yet non trivial aspect of a quality model is the relationship between internal and external quality. External quality, as seen by the customer, is what the system has to achieve. Internal quality, however, is what is visible during development. Therefore, making inferences between internal and external quality (attributes) is the key to evaluating quality early in the development process and controlling it appropriately.

In addition to linking internal to external quality, it is also useful to link internal quality determined early in the development process (e.g., design) to internal quality determined later in the development process (e.g., code or testing). While such a link does not allow to make inferences on the final product, it does make it possible to make inferences about quality that will be present at later stages of the development process.

The last type of relationship links a development process with its resulting product's quality. Understanding this kind of relationship enables a project manager to select processes that achieve the level of quality intended for each particular project. Relationships of this type are established by using empirical studies to determine the impact of individual development practices on product quality.

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 17 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

An additional important aspect regarding quality relationships is the context and environment in which each relationship was identified, since a relationship can stop being valid as soon as its original context is altered. A transfer of relationships to new environments must be performed carefully, by validating the transferred relationships in the new context. One way to understand this problem is by observing that context and environment also characterize a set of variables that affect quality and the interactions among its impacting factors. Changing the context and environment might also change these factors and thus the resulting level of quality.

## 2.5 MEASUREMENT

As pointed out in Section 2.3, the main objective of a quality model is to operationalize the term quality, by making it eventually measurable. For this reason, refinement of quality characteristics inside a quality model stops at the level of quality attributes, which are measurable properties of products, processes, or resources. In order to measure the property or attribute of an entity (i.e., process, product or resource) via **measurement**, numbers or symbols are assigned to its attribute (Fenton and Pfleeger, 1996; Agresti et al., 2002). This assignment has to follow clearly specified rules so that different people assign the same measurement values to the attribute.

A **measure** is “the number or symbol assigned to a quality attribute of an entity by making a measurement” (ISO 9126). There are **direct measures** that are determined by directly analysing the entity under study. **Indirect measures**, on the other hand, are derived from the measures of one or more other attributes.<sup>1</sup>

Each measure also needs to specify a **tolerance range**; that is, criteria for acceptable (“good”) values of the measure.

A **descriptive quality model** is an operational rule that determines how to measure an entity's attribute. Formally, a descriptive quality model can be defined as a function  $f$  that computes a measure  $\mu=f(x_1, \dots, x_n)$ , from the values of the measures  $x_i$ . (Briand et al., 1997). For example,  $\mu$  could be a model for defect density, which is defined as the number of defects found over product size. Here,  $x_i$  are the number of defects and size.

## 3 QUALITY MODELS STATE OF THE ART

This section presents the definitions of robustness and evolvability from the state of the art.


### 3.1 DEFINITION OF EVOLVABILITY AND ROBUSTNESS FROM F/OSS ASSESSMENT METHODOLOGIES

Several Methodologies to help select appropriate F/OSS components or products have surfaced in the past couple of years. The most known are QSOS, Open Source Maturity model (OSMM) by Gap Gemini, and OpenBRR. These methodologies have all create a evaluation template for scoring open source projects on different criteria. In this part of our work, we are only interested in inventorying the quality characteristics used by evaluated by these models but not the actual scoring system provided by these models. The work that specifies how to measure quality characteristics is left for later.

#### 3.1.1 QSOS

QSOS (QSOS, 2006) split its evaluation template in two kinds of sections: one generic section and several sections specific to a particular family of applications such as groupware, cms, database, etc. The specific section is related to the functionality offered. QUALOSS is not concerned with the *functionality* quality characteristic in turn, we are only interested in the generic section of the evaluation template. Figure 9 shows the QSOS structure.

<sup>1</sup> In this sense, a descriptive quality model is an indirect measure that measures a quality attribute

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 18 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

Below is the template of characteristics found in QSOS's generic template. We note that this template comes from the paper version (.pdf) and not the web version of the template, which is slightly different. In particular, the web version lists the item "Independence of development" in the "Industrialized Solution" category whereas the paper version includes it in the "Intrinsic Durability" category. Second, the web version elevated "Exploitability" as a top-level category where as the paper version includes it as a sub-category of the "Industrialized Solution" category. Finally, the paper version has a section on risk related to "Service Providing" that does not exist in the web version.

In some case, the terminology used by QSOS is not detailed enough, in such cases, we provide additional information in parenthesis, for example, Intrinsic Durability .. Adoption .. References is not precise enough as its is unclear to the reader what characteristic of reference are under consideration. In turn, we peeked in the scoring system to identify what characteristic(s) of References is(are) used in the scoring. In this case, it is the mission criticality of the system running the F/OSS product that is assessed. So, we add that information in parentheses.

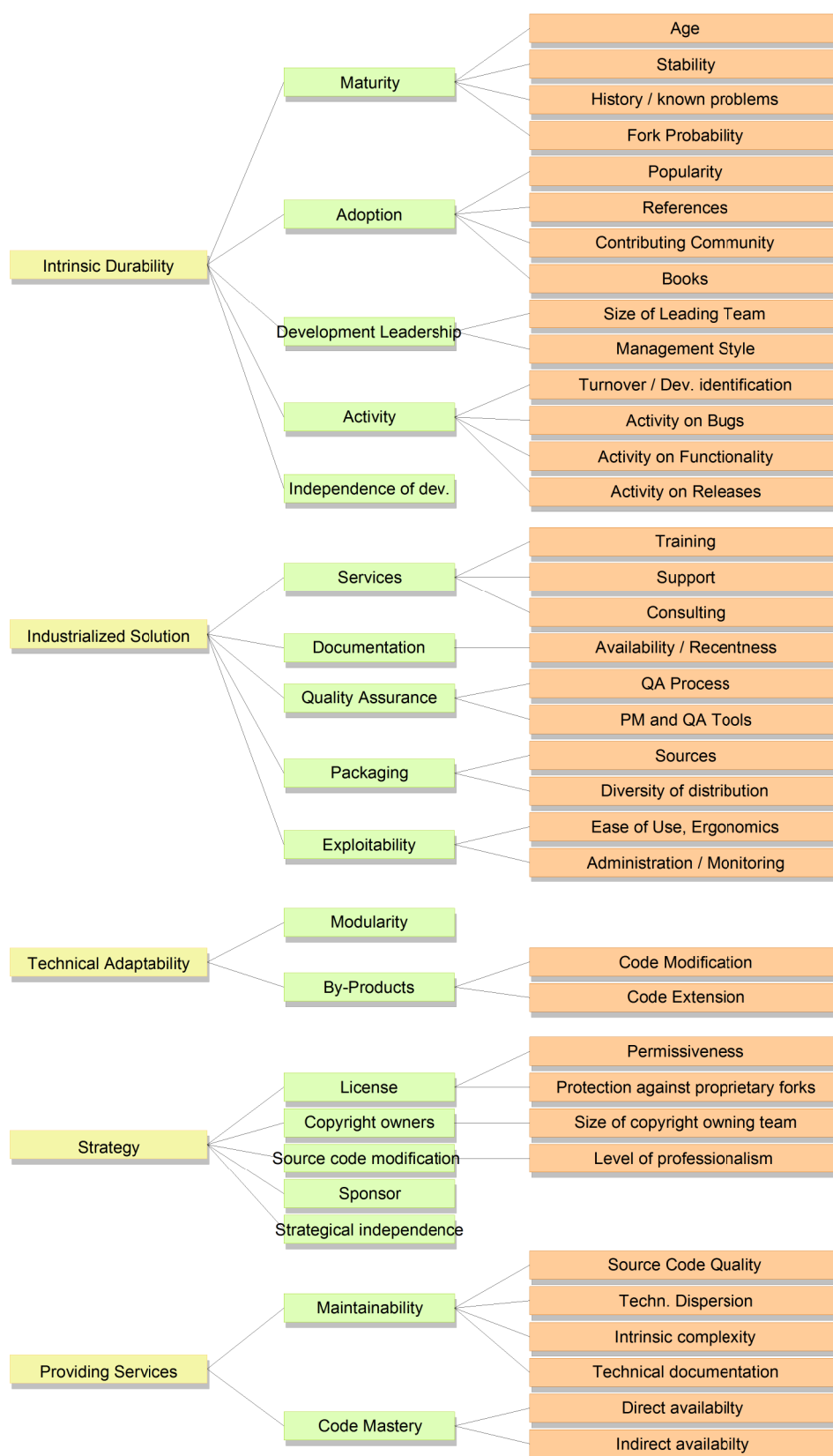



Figure 9: Generic criteria from QSOS version 1.6

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 20 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

### 3.1.2 Open Source Maturity Model (Cap Gemini)

Below is the hierarchy proposed by OSMM, as defined by Cap Gemini (Duijnhouwer and Widdows, 2003):

- Product
  - Age
  - Licensing
  - Human hierarchies
  - Selling Points
  - Developer community
- Integration
  - Modularity
  - Collaboration with other products
  - Standards
- Use
  - Support
  - Ease of Deployment
- Acceptance
  - User community
  - Market penetration

The hierarchy is much lighter and it is also very imprecise in its wording. For example, “Integration .. Standards” does not specify what characteristics related to standards are under consideration. To find that information, we peek in the scoring system. This reveals that the true characteristic being evaluated is the *recency of the standards followed*. We repeat this procedure to the other imprecise items of the OSMM hierarchy. The reworded hierarchy, presented in Figure 10 will make our comparison with QSOS and OpenBRR much more accurate.

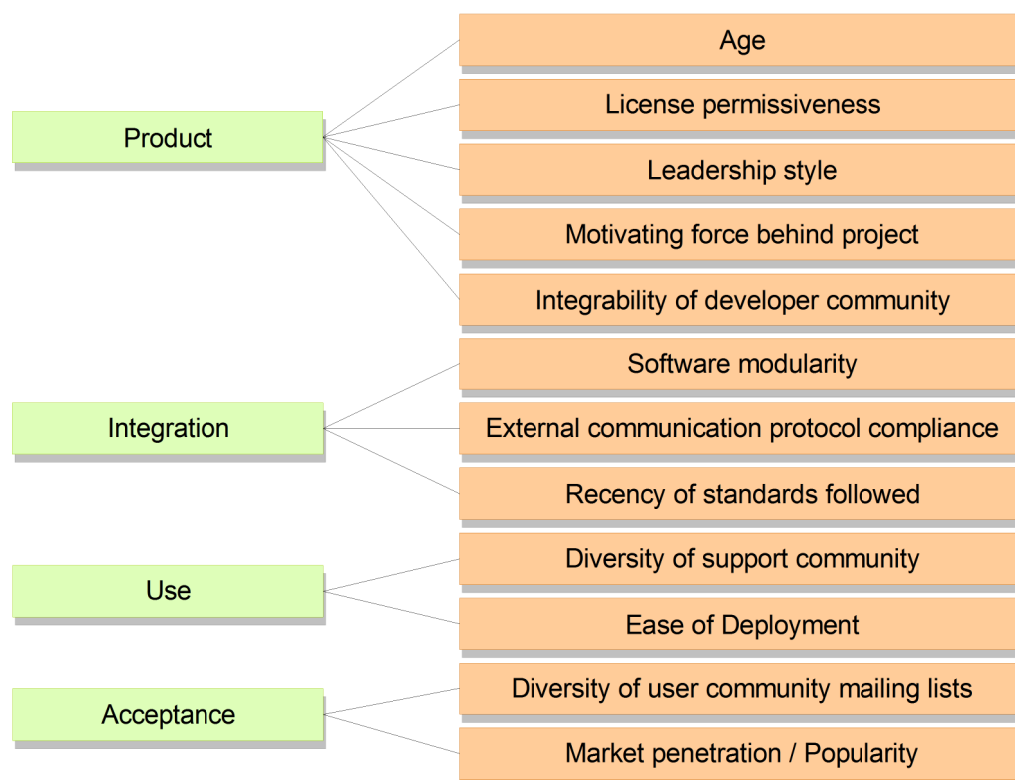



Figure 10: Open Source Maturity Mode


	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 21 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

### 3.1.3 Open Business Readiness Rating (OpenBRR)


The OpenBRR (OpenBRR; 2006) builds on two existing **maturity models**, Navica's **Open Source Maturity Model** (Golden, 2004) and Cap Gemini's equivalent (Duijnhouwer and Widdows, 2003). The OpenBRR hierarchy of quality characteristics looks as follows:

- Usability
  - End user UI experience
  - Time for setup pre-requisites for installing open source software
  - Time for vanilla installation/configuration
- Quality
  - Number of minor releases in past 12 months
  - Number of point/patch releases in past 12 months
  - Number of open bugs for the last 6 months
  - Number of bugs fixed in last 6 months (compared to # of bugs opened)
  - Number of P1/critical bugs opened
  - Average bug age for P1 in last 6 months
- Security
  - Number of security vulnerabilities in the last 6 months that are moderately to extremely critical
  - Number of security vulnerabilities still open (unpatched)
  - Is there a dedicated information (web page, wiki, etc) for security?
- Performance
  - Performance Testing and Benchmark Reports available
  - Performance Tuning & Configuration
- Scalability
  - Reference deployment
  - Designed for scalability
- Architecture
  - Is there any 3rd party Plug-ins
  - Public API / External Service
  - Enable/disable features through configuration
- Support
  - Average volume of general mailing list in the last 6 months
  - Quality of professional support
- Documentation
  - Existence of various documentations.
  - User contribution framework
- Adoption
  - How many books does amazon.com gives for Power Search query: "subject:computer and title:component name"
  - Reference deployment
- Community
  - Average volume of general mailing list in the last 6 months
  - Number of unique code contributor in the last 6 months
- Professionalism
  - Project Driver
  - Difficulty to enter the core developer team

Unlike the other two previous hierarchies, elements of OpenBRR hierarchy are very specific, for example, under the Quality category, we find "number of minor releases in the past 12 months". This is not a quality characteristic but as OpenBRR calls it: a metric. So the current OpenBRR hierarchy is not in a suitable form to compare it with the other two models. However, the metrics used by OpenBRR can be abstracted into

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 22 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

quality characteristics. For example, under Usability, the two metrics “Time for setup pre-requisites for installing open source software” and “Time for vanilla installation / configuration” can be abstracted to “Ease of Vanilla Deployment”. We repeat the same analysis for the different set of metrics found under every OpenBRR category. However, we moved information of two categories, in particular, characteristic extrapolated from Scalability migrated under *Architecture* and characteristic of documentation were moved under the *Support* category. The results of our abstraction are presented below in Figure 11.

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 23 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

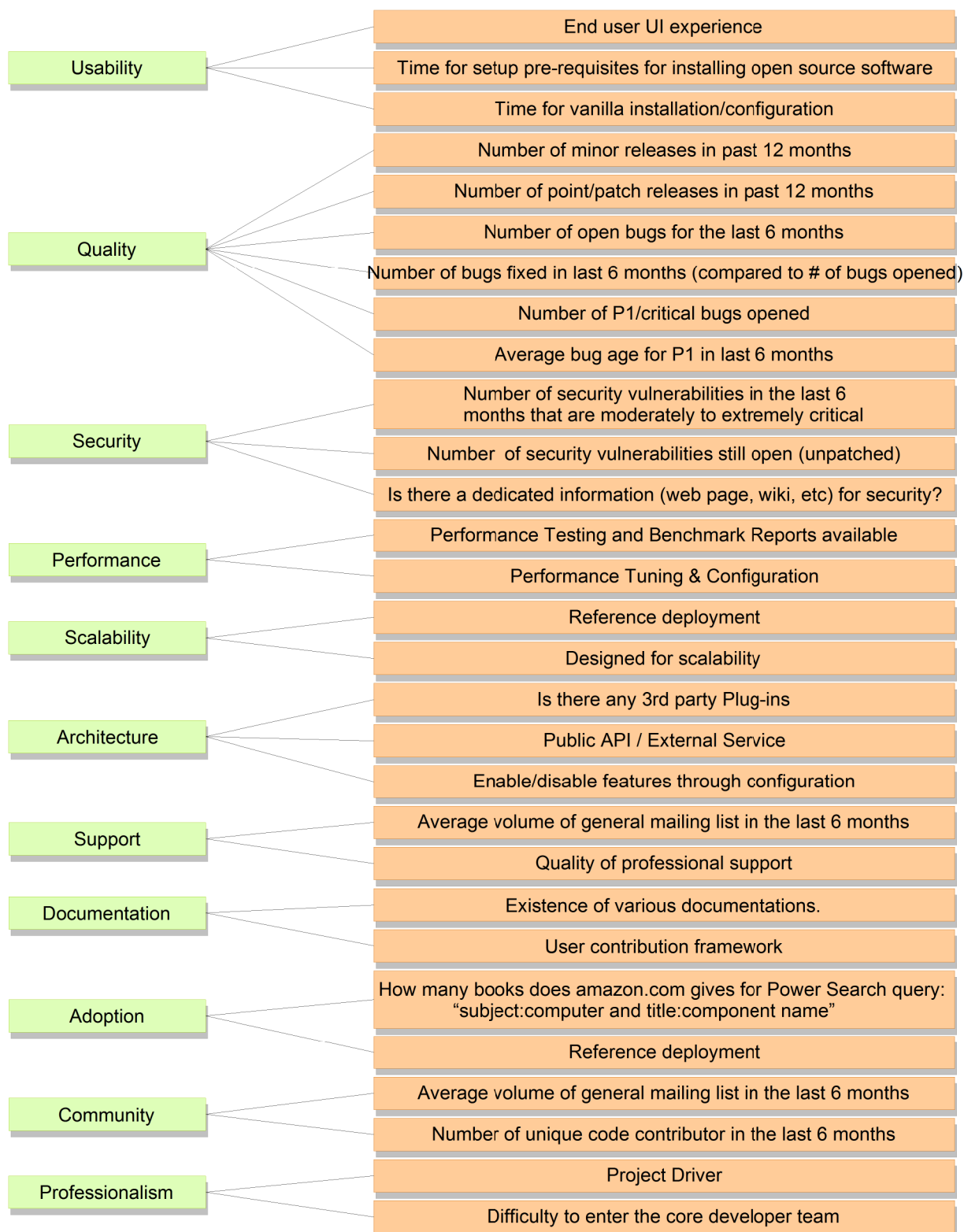



Figure 11: Open Business Readiness Rating

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 24 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

### 3.1.4 Comparing QSOS, OSMM and OpenBRR

This section compare the hierarchy of QSOS, OSMM and OpenBRR with the indent to later combine their quality characteristics in a single more comprehensive hierarchy of quality characteristics. In other words, our comparison is not to intended to find out which of the three model is better but rather to extract quality characteristics that are different between the three models. We will later classify these characteristics into a more comprehensive hierarchy.


We emphasize that the comparison is performed on our reworked versions of OSMM and OpenBRR, only QSOS was left changed.

Given that QSOS is the most comprehensive of the three models, we use it as the starting reference for our comparison. In particular, we perform a pairwise comparison between each QSOS leaf characteristic with each leaf characteristic of OpenBRR and OSMM. After, we enumerate the quality characteristics of OSMM and OpenBRR not covered by QSOS.


The possible results of a pairwise comparison between two characteristics A and B are:

- The two characteristic are equivalent ( $A = B$ )
- One characteristics includes the other ( $A < B$  (A is included in B) or  $A > B$  (A includes B)),
- The two characteristics have some similarity relationship of a fuzzy nature ( $A \sim B$ )
- The two relationship have nothing in common (not listed in the table)


In the comparison table below, the left column enumerate all QSOS characteristics and then, for every leaf characteristic of QSOS, we indicate whether OpenBRR and OSMM have corresponding characteristic with one of the relationship signs identified above ( $=$ ,  $<$ ,  $>$ , or  $\sim$ ). This is shown by the relationship sign preceding the characteristics in the OpenBRR and OSMM columns.

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 25 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

QSOS			OSMM	OpenBRR
Intrinsic Durability	Maturity	Age	= Product .. Age	NONE
		Stability	NONE	NONE
		History, known problems (= Management Ability)	NONE	NONE
		Fork probability, source of forking	NONE	NONE
	Adoption	Popularity	= Acceptance .. Current Market Penetration	~ Adoption .. Diversity of Deployments (related to diversity of user community)
		References (= mission criticality of references)	NONE	NONE
		Contributing community (= volume and diversity of community contribution)	> Acceptance .. Diversity of user community mailing lists	> Community .. Activity on mailing lists  > Community .. Size of Team Contributing Code (in past 6 months)
		Books (number of books published about products)	NONE	= Adoption .. Book Availability
	Development leadership	Leading Team (= Size of leading team)	NONE	NONE
		Management style (= level of democracy of management)	= Product .. Leadership style	NONE
	Activity	Developers identification, turnover	= Product .. Integrability of developer community	~ Community Professionalism .. Integrability / permeability of core developer team
		Activity on bugs	NONE	= Quality .. Reactivity of developer community to product problems .. Reactivity on critical bugs and Reactivity on all bugs
		Activity on functionalities	NONE	NONE
		Activity on releases	NONE	= Quality .. Release Activity (in past 12 months)
	Independence of development		~ Product .. Leading Style	~ Community Professionalism .. Independence of Management Community


	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 26 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

QSOS			OSMM	OpenBRR
Industrialized Solution	Services	Training (Diversity in geographical, cultural and gradual aspects)	NONE	NONE
		Support (Level of commitment assigned to support)	NONE	~ Support .. Professionalism of support
		Consulting (Diversity in geographical and cultural aspects)	NONE	NONE
	Documentation (Availability and recentness of documentation)		NONE	~ Documentation .. Documentation diversity
	Quality Assurance	Quality Assurance Process	NONE	NONE
		PM and QA Tools	NONE	NONE
	Packaging	Sources	NONE	NONE
		*nix packaging	NONE	NONE
	Exploitability	Ease of use, ergonomics	~ Use .. Ease of deployment	> Usability .. (ease of) Vanilla Deployability
		Administration/Monitoring (Availability of functionality for administration and monitoring)	NONE	NONE
Technical adaptability	Modularity (Software modularity)		= Integration .. Software Modularity	< Architecture .. Scalability (of design)
	By-Products	Code modification (Ease of build-ability)	NONE	NONE
		Code extension (Extensibility or plug-ability)	~ Integration .. External communication protocol compliance	~ Architecture .. Extensibility

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 27 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

QSOS			OSMM	OpenBRR
Strategy	License	Permissiveness	= Product .. License permissiveness and alternatives	NONE
		Protection against proprietary forks	~ Product .. License permissiveness and alternatives	NONE
	Copyright owners (Size of copyright owning team)		NONE	NONE
	Modification of source code (Level of professionalism of procedure for proposition of modification.)		< Product .. Integrability of developer community	~ Community Professionalism .. Integrability / permeability of core developer team
	Roadmap (availability + precision of the roadmap)		NONE	NONE
	Sponsor (Driving force behind product)		= Product .. Motivating force behind project	= Community Professionalism .. Independence of Management Community
	Strategical independence		NONE	~ Community Professionalism .. Independence of Management Community
Providing Services	Maintainability	Quality of Source Code (Volume of comment and use of design pattern)	NONE	~ Architecture .. Scalability ~ Performance .. Performance Tuning & Configuration (on user's end) ~ Architecture .. Extensibility
		Technological dispersion (number of prog.lang. used)	NONE	NONE
		Intrinsic complexity (Complexity of algorithms)	NONE	NONE
		Technical documentation (Design and arch doc + others)	NONE	~ Documentation .. Documentation diversity
	Code Mastery	Direct availability (Number of experts available within a consulting company)	~ Use .. Diversity of support community	~ Support .. professionalism of support
		Indirect availability (Number of experts available in partner companies of serv. prov.)	~ Use .. Diversity of support community	~ Support .. professionalism of support

Overall, the OSMM criteria are well covered by the QSOS model, in particular, 11 of the 12 OSMM criteria are covered by QSOS criteria. The only non-covered criterion of OSMM is *Recency of standards followed*. In the 11 criteria covered, 7 identified as exact matches. In the 4 remaining cases, 3 are only approximate (~) matches and the fourth one, "Diversity of user community mailing lists", is identified as being included (<) in the criteria "Adoption .. Contributing community" of QSOS.

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 28 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

QSOS and OpenBRR have less coverage in common than QSOS and OSMM. Our transformed version of OpenBRR has 24 leaf characteristics (compared to the 41 leaf characteristics of QSOS). Only 5 exact matches, 4 include relationships are identified. 13 similarity relationships are also identified but only 6 of these 13 cover QSOS characteristic that are not already covered by an exact match or an include relationship. In turn, 15 OpenBRR (4 + 5 + 6) are covered by QSOS and 9 are not. The uncovered characteristics are: “Usability .. UI learnability & Understandability”, “Quality .. Bug Reporting Activity (in past 6 months), all Security characteristics (“Reporting activity on product vulnerability”, “Reactivity on security bugs”, and “Community involvement on security issues”), “Performance .. Testability for performance (including presence of test suite and benchmark reports), “Architecture .. Feature configurability (on user's side)”, “Support .. Community reactivity to support questions”, and “Support .. Diversity of documentation contributors”.

We could perform a complete comparison between OSMM and OpenBRR however, we have most results available in the table above, the only completely missing relationships are those between OSMM and OpenBRR characteristics not covered by QSOS. This case is covered next.

The only non-covered characteristics of OSMM is “Recency of standards followed”, none of the OpenBRR cover this characteristics.

When building our comprehensive hierarchy of quality characteristics, we can use the results of our comparison as follows.

- When characteristics are exact matches, only one of them needs to be inserted in our comprehensive hierarchy. (we are left to choose the characteristic with the most adequate terminology)
- When two characteristics have an include relationship ( $A < B$ ), then the comprehensive hierarchy may decide to include both showing a parent-child relationship (where B is the parent of A) or it may also decide to omit A, if B is already considered low-level enough. B is considered low-level enough if we can foresee a metric formula to estimate B.
- When characteristics have similarity relationships ( $\sim$ ), the procedure is not as systematic. We must consider each characteristic involved in the relationship and decide whether to include in the same sub-tree of the hierarchy or whether to include them in different sub-trees of the hierarchy (if possible, the similarity ( $\sim$ ) relationship should be kept explicit through a cross referencing mechanism, for example). A characteristic in a “similarity” group may also be eliminated from the final hierarchy if it is completely covered by other characteristics in that group. For example, in a group of characteristics A, B, and C, the subgroup A and B cover the characteristic C in such case, C can be either eliminated.

### 3.2 DEFINITION OF EVOLVABILITY AND ROBUSTNESS FROM THE STATE OF THE ART AND PRACTICE

This section describes several quality models that represent the state of the art and practice in quality modelling (Freimut et al., 2003; Hartkopf et al., 2007).

#### 3.2.1 The Historic Models: McCall and Boehm

Two early models, originally proposed by McCall (McCall et al., 1977) and Boehm (Boehm et al., 1973; Boehm et al., 1976) are shown in Figure 12 and Figure 13.

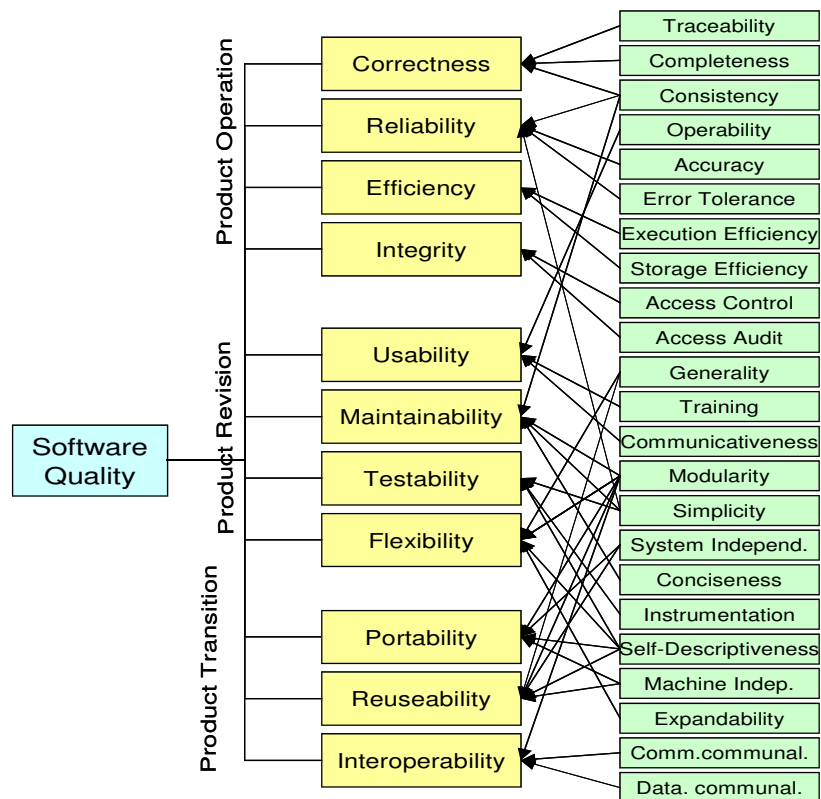


Figure 12: The product quality model proposed by McCall

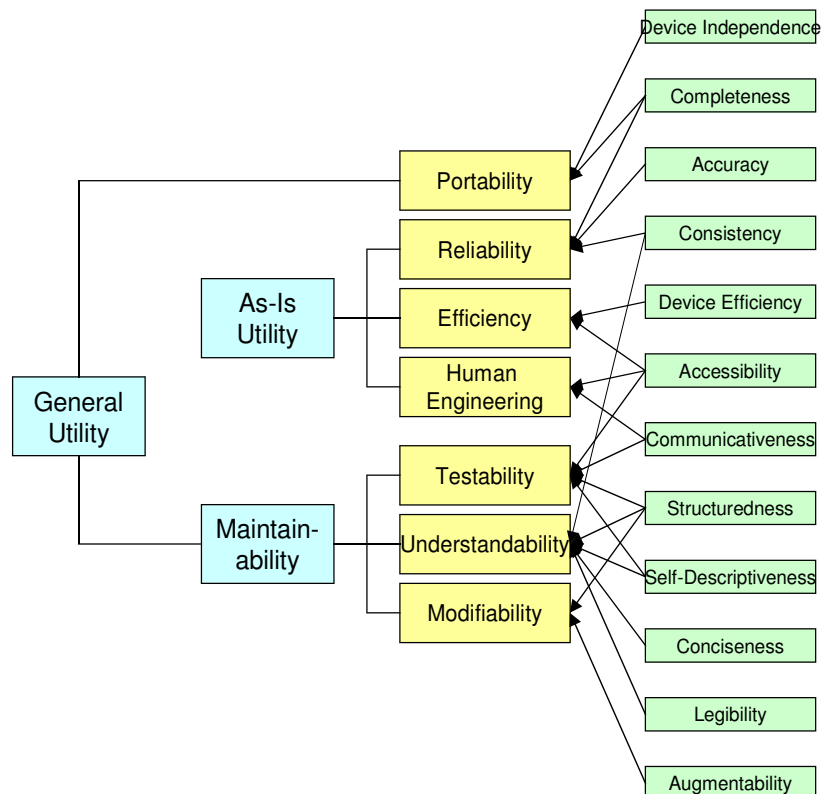



Figure 13: The product quality model proposed by Boehm

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 30 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

Both models identify key characteristics of quality from a user perspective. These key characteristics (called factors, synonymous with quality characteristics) are mostly high-level external characteristics (e.g., reliability, maintainability) but can also be internal ones (e.g., testability). These high-level characteristics are refined into sub-characteristics called criteria, which represent an internal view on quality. These criteria are supposed to influence the achievement of a key characteristic. Measures are used to measure the criteria.

From the QualOSS viewpoint; that is, for robustness and evolvability, the following sub-characteristics are relevant:

- In McCall's model: reliability, efficiency, integrity, usability, maintainability, testability, flexibility, portability, reusability, and interoperability.
- In Boehm's model: portability, reliability, efficiency, testability, understandability, and modifiability

### 3.2.2 Quality Model of the Deutsche Gesellschaft für Qualität

In the early 1980s the German Association for Quality (Deutsche Gesellschaft für Qualität e.V.(DGQ)) formed a working group to come up with a recommendation on how quality for software can be achieved. The group consisted of employees among others from Nixdorf Computer, IBM Germany, Siemens AG, or Fraunhofer Institute for Production Technology. This initiative was meant to contribute to the observation that software increasingly permeated all kinds of products and that the same quality rules should be applied for software as for any other industrial good. The result of this effort was published in 1986 (DGQ, 1986).

The focus of the work is how to build up, organize, and run an entire software quality assurance organization. Principles, activities, methods, tools of software quality assurance are describes as well as economic and legal aspects. Furthermore, they came up with a model that distinguishes between software quality characteristics for software programs and software documentation.

The quality model defines twelve quality characteristics for software programs. Figure 14 depicts the quality characteristics (yellow) and if refined the sub-characteristics (green). Every quality characteristic is defined and described as far as possible with respect to sub-characteristics, metrics, constructive activities to achieve the quality, testing methods, scope of testing, documentation of testing, impact of characteristic to quality costs, quality documentation, application context of characteristic, and mutual impact to other quality characteristics. Table 2 and Table 3 give an example for the quality characteristics "maintainability" and "robustness", respectively.

From the QualOSS viewpoint, the following characteristics are relevant: adaptability, usability, efficiency, maintainability, portability, robustness, safety/security, interoperability, reusability, and reliability.

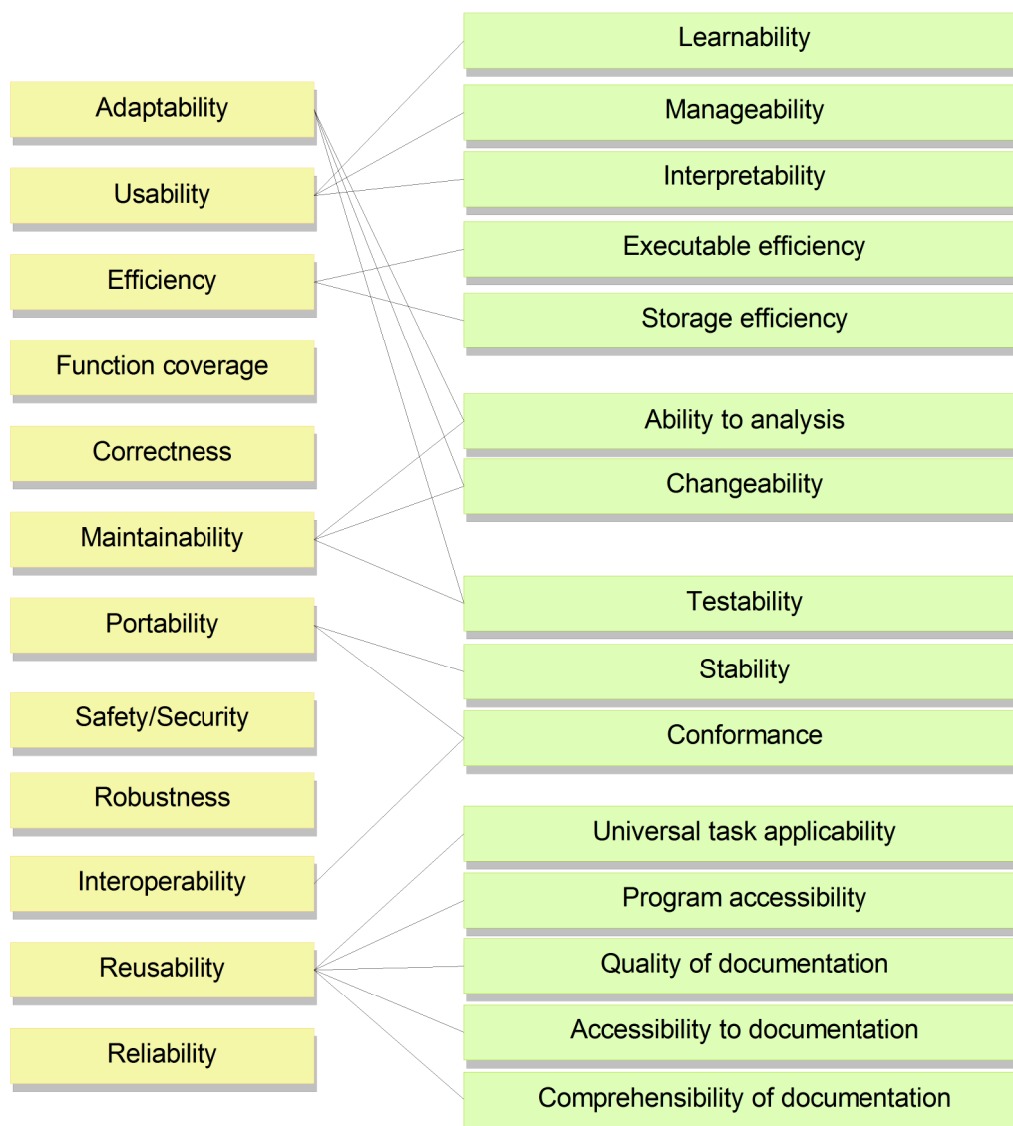




Figure 14: DGQ Quality Model for software programs

Aspect	Description
Definition: <b>Maintainability</b>	Suitability for detecting and removing faults (i.e., causes of failures).
Sub-characteristics	Learnability, manageability, interpretability
Metrics	In general, through average (or total) amount of time and knowledge required for maintenance tasks under given constraints.
Constructive activities	<ul style="list-style-type: none"> <li>- Guidelines for design and coding</li> <li>- Naming conventions</li> <li>- Rules for maximum complexity of modules</li> <li>- Guidelines for documentation- ....</li> </ul>

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 32 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

<b>Aspect</b>	<b>Description</b>
Testing methods	<ul style="list-style-type: none"> <li>- Checklists</li> <li>- Collect metrics (effort of maintenance tasks)- ...</li> </ul>
Scope of testing	The entire program including documentation must be tested.
Documentation of testing	The testing must be documented.
Impact of characteristics to quality costs	Additional costs for appropriately documenting and structuring the system can be high. However, these additional costs are usually lower than expected costs caused by low maintainability.
Quality documentation	Regular reports on <ul style="list-style-type: none"> <li>- failures / faults</li> <li>- reffort and qualifications for fault removal- ...</li> </ul>
Application context	High maintainability is usually important for software with high availability requirements (operating systems etc.). Company-specific reasons can also demand for high maintainability.
Mutual impact to other characteristics	<ul style="list-style-type: none"> <li>- High maintainability increases adaptability, portability, reusability, correctness, and reliability of the system.</li> <li>- Maintainability can have a negative influence on efficiency, depending on the selected implementation technique.</li> </ul>

Table 2: Description of the characteristic "maintainability"

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 33 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

<b>Aspect</b>	<b>Description</b>
Definition: <b>Robustness</b>	Suitability of a program to show correct/defined behavior under erroneous conditions (hardware, input, or runtime errors).
Sub-characteristics	Not refined into sub-characteristics
Metrics	<p>Usually specified in terms of input errors, e.g.,</p> $R_E = \frac{\text{Number of checked inputs}}{\text{Total number of inputs}}$ <p>Inputs, in this case, include explicit input data as well as parameters. Total number of inputs also includes data outside of the definition domain. If the program responds correctly to those, <math>R_E</math> approaches 1. This is to be seen as heuristic, as it is usually not possible to prove <math>R_E</math> for all possible inputs.</p>
Constructive activities	<ul style="list-style-type: none"> <li>- Plausibility check for each input datum and parameter</li> <li>- Rollback for write operations</li> <li>- System recovery in case of system errors</li> <li>- Using existing data to estimate missing data; for incorrect data, use self-correction</li> <li>- Check hardware components for availability and correctness.</li> </ul>
Testing methods	<ul style="list-style-type: none"> <li>- Checklist for existence of constructive activities</li> <li>- test with incorrect data (i.e., data outside of definition domain)</li> </ul>
Scope of testing	<p>The entire program must be tested, however, exact scope depends on goal of test:</p> <ul style="list-style-type: none"> <li>- detect all faults --&gt; 100% coverage needed</li> <li>- determine quality of system, find and remove faults --&gt; (random) samples may be sufficient</li> </ul>
Documentation of testing	The testing must be documented.
Impact of characteristics to quality costs	Costs for reaching appropriate robustness can be high. However, failure costs decrease (e.g., for system breakdown).
Quality documentation	-
Application context	High maintainability is usually important for software with intensive user interactions, systems with high availability requirements (e.g., process/workflow software), systems that operate in noisy environments. Company-specific reasons can also demand for high maintainability.
Mutual impact to other characteristics	<ul style="list-style-type: none"> <li>- High robustness increases usability, safety, and reliability of the system.</li> <li>- Robustness decreases efficiency, as additional checks are necessary.</li> </ul>

Table 3: Description of quality characteristic "robustness"

### 3.2.3 FURPS/FURPS+

The FURPS/FURPS+ model was developed at Hewlett Packard (HP) to improve the quality of their products (Grady and Caswell, 1987). HP employs FUPRS+ as a means to drive customer satisfaction goals by supporting the identification of measurable product metrics (Grady, 1992).

FURPS+ defines five quality characteristics (**Functionality**, **Usability**, **Reliability**, **Performance**, **Supportability**), which are divided into 27 sub-characteristics (see Figure 15). The authors do not describe how they obtained their model.

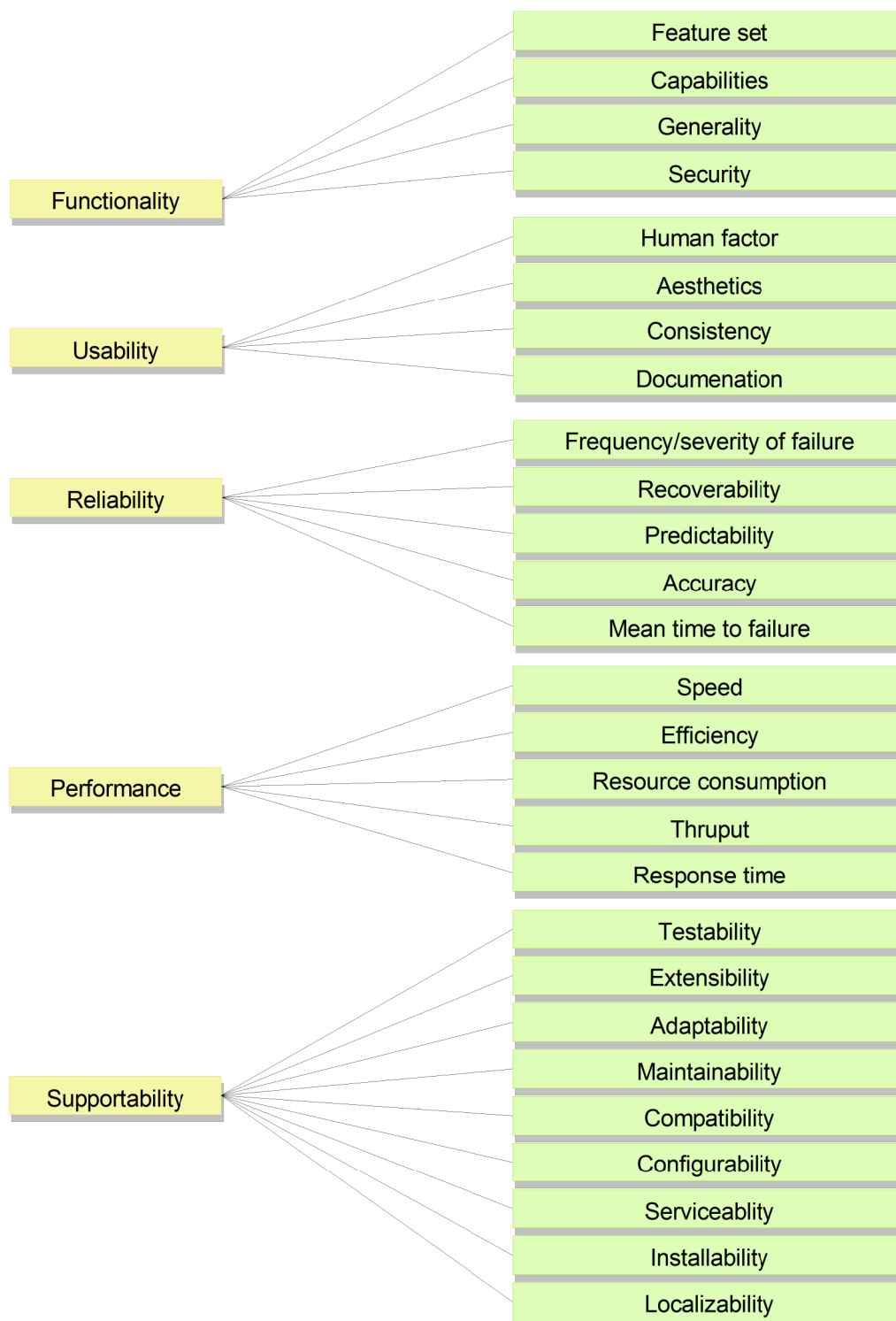




Figure 15: FURPS+ Quality Model

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 35 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

Each of the quality criteria maps to one or more metrics. The FURPS model is used to simplify the process of defining the appropriate measurements in each life cycle phase (Specification / Design / Implementation / Testing / Support), through focusing on specific quality characteristics.

Using the FURPS model involves two steps: establishing priorities and making quality attributes measurable. Establishing priorities is important because of the trade-offs involved between quality factors. Through addition of a new function, functionality might be improved but performance, usability, and/or reliability decreased. Thus the decision has to be made, what type of quality is relevant in this project and a prioritized list of quality factors has to be defined. At this point of the project it should be made clear what is really important and how the priorities in quality affect cost and complexity of the product. Further a ranking of the quality criteria for the top quality factors can be performed.

Once priorities have been established, measurable goals for each quality factor have to be defined. These measures are, as the priorities, project specific and also depend on the phase of the life cycle. Table 4 lists some FURPS+ metrics through the whole product life cycle for usability, reliability, and supportability, which are FURPS' relevant characteristics for QualOSS.

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 36 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted


<b>Life cycle Phases</b>	<b>Specification</b>	<b>Design</b>	<b>Implementation</b>	<b>Testing</b>	<b>Support</b>
<b>Usability</b>	# target users to review specification or prototype % grade on documentation plan by target users % grade on usability of prototype	% grade of design as compared to objectives # changes to prototype manuals after review	% grade by other lab user % grade by product marketing, documentation % original users to review any change	# changes to product after alpha test % grade from usability lab testing % grade by test sites	# user misunderstandings
<b>Reliability</b>	# omissions noted in reviews of objectives # changes to project plan, test plan after review	# changes to design after review due to error % grade of design as compared to objectives	% code changes due to reliability errors discovered in reviews % code covered by test cases # defects/KNCSS during module testing	MTTF (MTBF) % hrs reliability testing # defects/1 K hrs # defects total defect rate before release checkpoints	# known problem reports # defects/KNCSS
<b>Supportability</b>	# changes to support after review by field & CPE	# design changes by field & CPE # diagnostic/recovery changes by field & CPE input	MTTR objective (time) MTTC objective (time) time to train tester, use of documentation		

Table 4: Excerpts of metrics to be measured in different life cycle phases with respect to performance

MTTF: Mean Time To Failure  
MTBF: Mean Time Between Failure  
MTTC: Mean Time To Change  
MTTR: Mean Time To Recovery  
CPE: Current Product Engineering  
KNCSS: Kilo-lines of NonComment Source Statements

The second approach on how to apply FURPS is given in (Grady, 1992). A three step procedure is proposed:

- Use the FURPS+ quality model (see Figure 6) as a checklist and refine the quality characteristics according to the Goal-Question-Metric paradigm. For example, identify the goal “optimize delivered product functionality” and ask the question “What functions do the customers need now?”

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 37 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

- Select metrics that help to know how well the product is satisfying customer's needs and wants. It is important to measure the progress toward final, measurable product goals throughout product definition and development. An example is shown for the quality characteristic "functionality" in Table 7.
- Include the characteristics in the quality plan and document how the quality characteristics respectively metrics are measured in every life cycle phase.

FURPS itself does not contain a technique to derive the metrics for the quality criteria. In general this derivation is context and project (customer) dependent, unless a static definition of quality is given. Goal oriented measurement approaches can help in finding the appropriate metrics. In praxis some metrics will be reused from project to project, though this is not recommended.

The perspective of the quality factors and the quality criteria is product and customer oriented. Thus FURPS, similar to ISO 9126, has a *product view* on quality, but user oriented. So FURPS is a loose integration of the *user* and the *product view*. In the context of embedded systems it should be mentioned that the FURPS model has been used in the development for software and for hardware/software systems. Experience showed that aspects of reliability and supportability were heavily influenced by hardware design decisions.

Even if little data is available HP claims that they could reduce the development cost and the number of errors per 1000 Lines of Code in the source code significantly (Balzert, 1998).

### 3.2.4 Quality Model of the NASA SATC

The NASA Software Assurance Technology Center (SATC) (Hyatt and Rosenberg, 1996) developed a quality model with the specific purpose of supporting project managers. The aim behind the selection of this particular viewpoint was to motivate NASA project managers to dedicate part of their project budgets to quality modelling. Tangible benefits such as increased information on the development process and its risks, increased confidence that the software will be usable when completed, and real cost savings such as decreased test time, were considered important motivators for this target group. Consequently, the software product quality attributes were defined based on corresponding project risks.

The SATC analyzed three fixed model approaches, namely those of McCall, Boehm, and ISO 9126/1993, but found them not to satisfy their needs. For this reason, they came up with their own quality model, which, in compliance with the ISO 9126/1993, distinguishes among quality goals, attributes, and metrics. The objectives of the model definition were, first, to determine an orthogonal set of attributes and metrics (i.e., each attribute and metric appears only once in the model), and second, to define metrics that can be based on objective process and product data, instead of, for example, expert assessments. The quality model is depicted in Figure 16.

Only some of the attributes identified in this model are relevant to the QualOSS project. In particular, attributes related to the goal "Product Quality and Risk" are specially relevant because they are directly related to source code. The following list briefly describes these attributes (descriptions were taken from (Hyatt and Rosenberg, 1996):

- Structure/Architecture: the evaluation of the constructs within a module to identify possible error-prone modules and to indicate potential problems in usability and maintainability.
- Reuse: the suitability of the software for reuse in a different context or application.
- Maintainability: the suitability of the software for ease of locating and fixing a fault in the program.
- Documentation: the adequacy of internal code documentation and external documentation.

The SATC report describes a number of approaches to measure these attributes directly on a system's source code.

The correctness attribute in goal “Testing Effectivity” is also relevant for our purposes. The measures proposed by SATC for this attribute are, however, more related to the project than to the product itself, and would have to be adapted to be applicable to F/OSS projects.

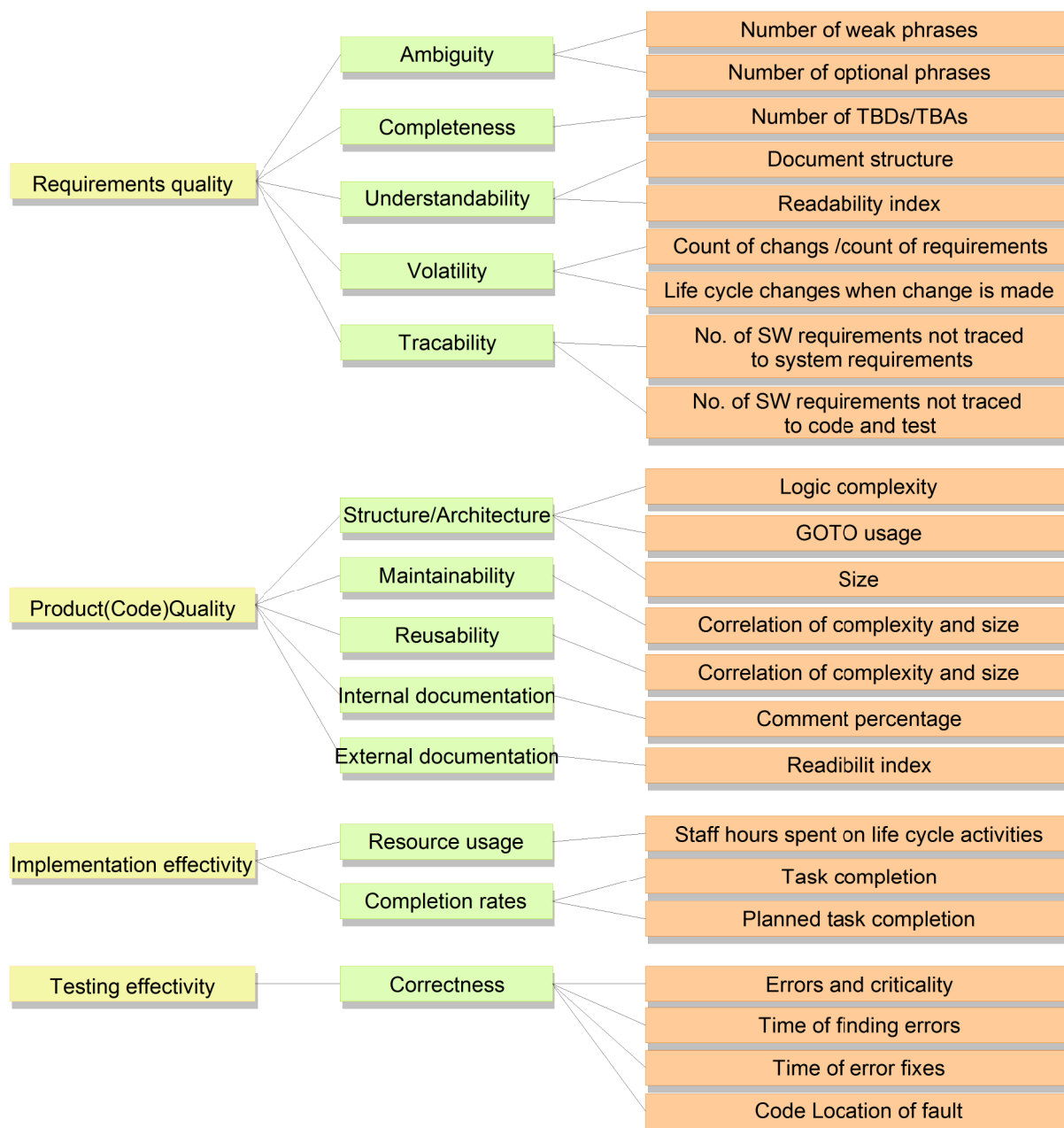



Figure 16: Quality Model of the NASA SATC

### 3.2.5 The Standard: ISO9126

The standard quality model as proposed in ISO9126 is shown in Figure 17. This model was developed in the early 1990s as an attempt to consolidate the many views of software quality (e.g., Boehm, McCall, FURPS). Its objective was to act as worldwide standard. In 2001, a revision was agreed. The newer version retains the same software quality characteristics as the older one. Major differences concern amongst other issues the introduction of normative sub-characteristics, the introduction of quality in use, or the co-ordination with the

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 39 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

content of ISO/IEC 14598, which defines process quality, which in turn contributes to improving product quality, and product quality contributes to improving quality in use.

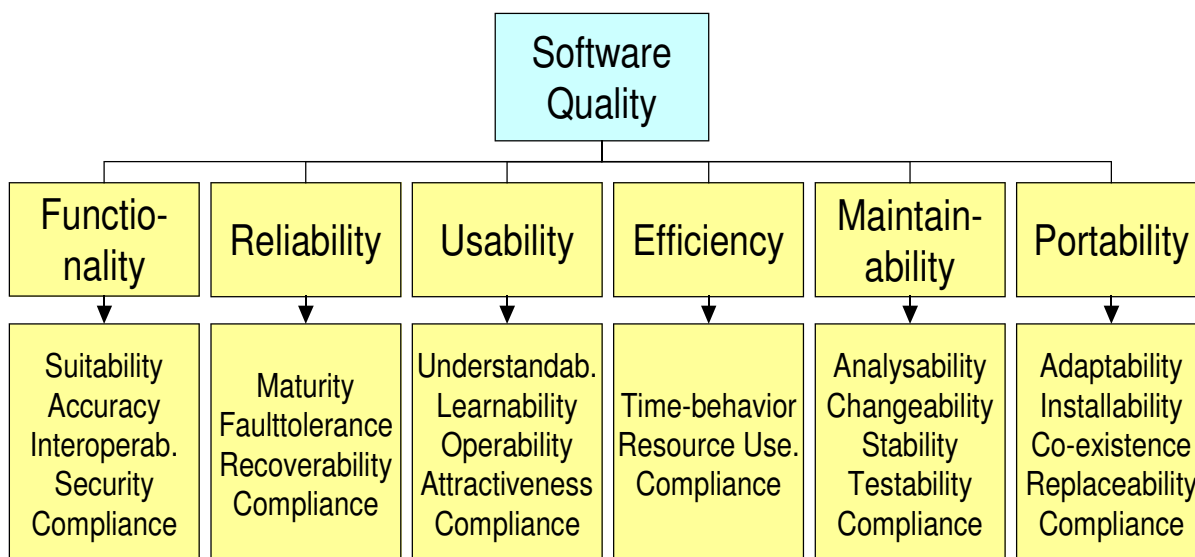


Figure 17: ISO 9126 Quality Model


The ISO 9126 quality model framework distinguishes between internal and external quality and quality in use. Internal quality is defined as “the totality of characteristics of the software product from an internal view”. In contrast to internal quality, external quality is defined as “the totality of characteristics of the software product from an external view. It is the quality when the software is executed, which is typically measured and evaluated while testing in a simulated environment with simulated data using external metrics.” Quality in use is defined as “the user’s view of the quality on the software product when it is used in a specific environment and a specific context of use. It measures the extent to which users can achieve their goals in a particular environment, rather than measuring the properties of the software itself”. The standard ISO 9126 is divided into four parts. The first part introduces the concept of product quality using quality models in general. Part two to four are Technical Reports, each one of them assigned to one quality type. Quality can be described with attributes<sup>1</sup>. Internal and external quality attributes are directly assigned to the software product, whereas quality in use attributes describe the effect of the software product in specific contexts of use.

### 3.3 Discussion

The quality models presented in Section 3.2.1 through Section 3.2.5 focus mostly on product measurement. On the other hand, existing F/OSS rating or assessment methodologies, presented in Section 3.1, focus on community measures and largely neglect product measurement. Moreover, they do not define their quality characteristics; that is, they can be said to be rather unsystematic. The QualOSS quality model has to take into account the whole project; that is, product and community aspects; thus, the approach taken in QualOSS has to be to bring these two worlds together.

## 4 RESULTS FROM INTERVIEWS

The goal of this section is to define the goal for the QualOSS quality models from the stakeholders' viewpoint. Therefore, we conducted a series of structured interviews to elicit usage scenarios for F/OSS software, and to elicit evaluation criteria for F/OSS software applied by stakeholders in practice.

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 40 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

## 4.1 USAGE SCENARIOS

Usage scenarios define how OSS will be used in organizations. In that way, usage scenarios define the organizational context and determine the goals of OSS evaluation. They will influence the used quality models insofar as different attributes or subcharacteristics will have a different importance (or weight); for example, analyzability plays a minor role if products are used “as is” (e.g., Apache), but when a F/OSS component has to be integrated into a company's product, readability is of crucial importance.

Usage scenarios will be defined through the survey by practitioners and by academia as well as consulting with all QUALOSS partners, especially, our Industrial partners, AdaCore and ZEA partners.

The resulting scenarios can be classified along two dimensions: Granularity of the F/OSS component, and the goal environment (see Figure 18).

The granularity of F/OSS component determines whether it is used as is (end-application), or whether it is used to build an application (platform level). For example, a webserver, such as Apache, is used to build web-applications on top of it, while office products, such as OpenOffice, are typically not modified and used as-is.

The goal environment for F/OSS component describes in which context the F/OSS component is intended to be used: Embedded, as part of an external service, internal service, as desktop application, or for development infrastructure. Obviously, if an F/OSS component is part of an external service, priorities of quality requirements (which would be mainly reliability) on it are different than when it is used as Desktop application (where usability plays a major role).

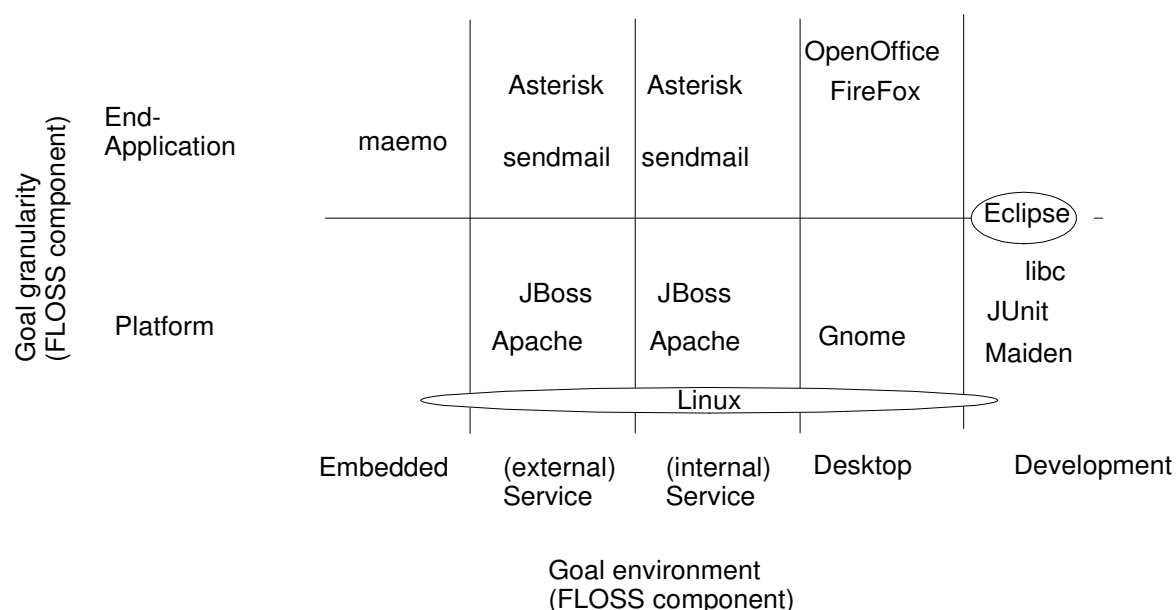



Figure 18: Structure of F/OSS usage scenarios

In addition to the categories of scenarios identified above, the **intended F/OSS usage**, **type of F/OSS component** and **product characteristics** of the product that integrates F/OSS components have to be considered.

Associated types of **intended F/OSS usage** include:

- Integrate an F/OSS product into a company's infrastructure (for example, using Apache as a web server for my website)

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 41 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

- Integrate an OSS product/components into a software product/system developed by a company (for example, using mysql as a part of a bug tracking system I develop)
- Fork an existing open source component (ex. Gforge forks from SourceForge)
- Extend an open source product to communicate with my product (independent of my product's license)
- Select an Open Source Language and Libraries to develop my product on (Ex. Using Python to develop an ERP or Ada for a air traffic control system)

#### Type of F/OSS components:

- A system (or product) integrates an open source Database technology (mysql, Postgres, ZODB, db4object, etc) or even an Object-Relational mapping technology such as Hibernate.
- A system (or product) integrates an open source Middleware (for distributed computing such as Zope, jboss, jonas, geronimo, etc or even grid computing, Globus, InnerGrid ) or a Distributed framework such as the Spring framework for java.
- A system (or product) integrates a visualisation libraries: Tk, SWT, dot and Graphviz, etc)
- A system (or product) integrates domain-specific computational libraries (such as financial computation modules or specific modules for matrix operations)

#### Product characteristics:

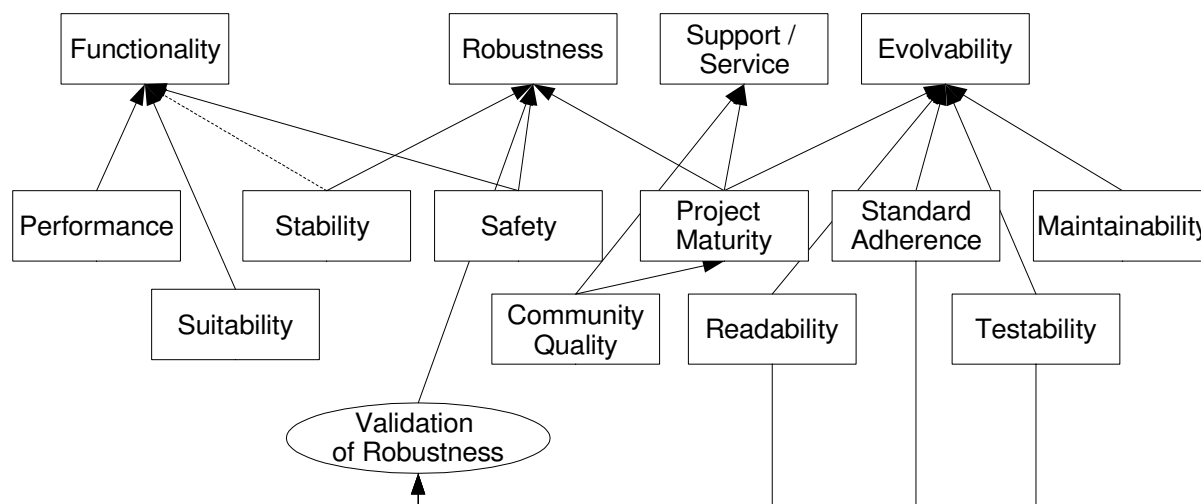
- The system (or product) that integrates OSS has safety critical issues associated with it
- The system (or product) that integrates OSS is embedded
- The system (or product) that integrates OSS is distributed
- The system (or product) that integrates OSS runs on a Grid
- The system (or product) that integrates OSS runs locally on a single machine


## 4.2 QUALITY MODEL FROM INTERVIEWS

The goal of the interviews was to probe the relevant quality aspects used by F/OSS users in industry when assessing F/OSS components. The approach we followed was to elicit goals from industrial partners through a structured interview. In total, we interviewed nine practitioners, in most cases IT responsables in their organizations, from five different domains: OSS developer/integrator, general IT, (web-based) Services, Health care, and public administration.

Preliminary results indicate that in all cases, the evaluation criteria for F/OSS components are ad-hoc; that is, usually one expert does the evaluation using (often implicit) criteria for evaluating required product qualities. In the interviews, we tried to elicit these criteria, as they can serve as basis for a definition of evolvability and robustness from the practitioners' viewpoint.

Figure 19 shows the preliminary results from the interviews concerning relevant quality criteria for evaluation of F/OSS components. There are four top-level constructs that users are interested in: Functionality, robustness, level of support, and evolvability.



	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 42 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

**Functionality:** Typical evaluation criteria are *performance* and *suitability* to the problem to be solved; in some cases, *stability* and *safety* are considered as part of the required functionality. These criteria are usually evaluated by conducting “ad-hoc” testing.

**Robustness:** Typical evaluation criteria are *stability*, *safety*, *maturity*, and *community quality*. In addition, *readability*, *adherence to standards*, and *testability* are considered as enhancing robustness, as they increase the evaluator's trust into the product.

**Support:** Evaluation criteria are *community quality*, and *maturity*

**Evolvability:** Typical evaluation criteria are *maturity*, *readability*, *adherence to standards*, *testability*, and *maintainability*.

In the following, we elaborate on the criteria used to evaluate robustness and evolvability.

**Community quality:** Evaluation criteria used are the continuity of the community (i.e., whether the project will go on further), activity in mailing lists, whether developers with reputation are in the team, the size of the community, whether company support exists for the community, quality of responses to questions / ability of the community to explain questions, and response time for bug fixes.

Continuity of the community is evaluated by looking at the frequency of releases, ease of updates, whether the project is based on standards, existence of a roadmap and evaluation of previous timeline, and the project is sufficiently focused to guarantee future survival.

**Project maturity:** This is evaluated by using criteria such as that the project should be “not too young” (i.e., age of the project), neither should it be a closed project, success cases should exist in large companies, a stable version of the product should be available, and user opinions and OSS community views should be generally positive.

**Stability and safety:** These criteria are evaluated by looking at user opinions in mailing lists, and by doing ad-hoc tests.

**Readability, maintainability and testability:** These criteria are evaluated by looking at the clarity of code, use of standards in the project, and at code documentation. That is, the interviewed persons do not distinguish between these different quality attributes, except by using different implicit criteria in testing.

**Standard adherence:** Main evaluation criterion is whether the project adheres to relevant development standards, such as design patterns or existing libraries. Concrete criteria used are typically defined ad-hoc by the evaluators.

The insights from the interviews will provide one input for defining a quality model for evolvability and robustness of F/OSS components. Another input is a literature review, which shall be detailed in the next section.


## 5 CONSOLIDATED MEASUREMENT REQUIREMENTS FOR QUALOSS

This section describes the goals derived from the usage scenarios. It also contains the consolidated QualOSS definition of robustness and evolvability in terms of quality characteristics and sub-characteristics. In terms of GQM, this would be the goals and questions of GQM. Metrics for the questions will be defined in task 1.3.

### 5.1 BUSINESS GOALS

Business goals can exist on different levels of hierarchy; on the lowest level, they represent software goals, i.e., business goals that explicitly refer to software development.

Business and software goals can be described using the following template:

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 43 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

<b>Name</b>	Name of the business goal.
<b>Description</b>	Narrative description of the business goal.
<b>Activity</b>	Activity that is supported with the business goal (e.g., improve, stabilize, check).
<b>Focus</b>	The focus of the business goal (e.g., cost, profit, turnover).
<b>Object</b>	The object to which the business goal relates (e.g., people, market, a project, a collection of projects).
<b>Quantification</b>	Quantification of the business goal (e.g., max. 10% budget overrun).
<b>Timeframe</b>	Timeframe in which the business goal is to be pursued (e.g. 5 years)
<b>Scope</b>	Scope or context of the activity (e.g., single development project, collection of processes, whole organization)

So far, we identified the following main business goals:


<b>Name</b>	Productivity
<b>Description</b>	Through F/OSS usage, the productivity will be increased, as much functionality is already available without having to implement it.
<b>Activity</b>	Improve
<b>Focus</b>	Productivity
<b>Object</b>	Project
<b>Quantification</b>	<unable to specify at the moment>
<b>Timeframe</b>	n/a
<b>Scope</b>	All projects in a company

<b>Name</b>	Time To Market
<b>Description</b>	Through F/OSS usage, the time to market will be reduce, as much functionality is already available without having to implement it.
<b>Activity</b>	Improve
<b>Focus</b>	Time to market
<b>Object</b>	Project
<b>Quantification</b>	<unable to specify at the moment>
<b>Timeframe</b>	n/a
<b>Scope</b>	All projects in a company

<b>Name</b>	Software Quality
<b>Description</b>	Through F/OSS usage, the quality of produced software will be increased, as (and if) the used F/OSS component already has a high quality.
<b>Activity</b>	Improve
<b>Focus</b>	Software quality
<b>Object</b>	Product
<b>Quantification</b>	<unable to specify at the moment>
<b>Timeframe</b>	n/a
<b>Scope</b>	All projects in a company

## 5.2 MEASUREMENT GOALS

Software goals can be refined into measurement goals, which specify concrete goals to be pursued by data measurement. Measurement goals can be described using the GQM template:

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 44 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

<b>Name</b>	Name of the measurement goal.
<b>Description</b>	Narrative description of the measurement goal.
<b>Business goals</b>	Name of related business or software goal.
<b>Object</b>	The object to which the measurement goal relates
<b>Purpose</b>	Purpose of the measurement/analysis (e.g., characterize, predict)
<b>Quality focus</b>	The object's quality attribute that is addressed by measurement.
<b>Viewpoint</b>	The viewpoint or role from which the analysis is done
<b>Context</b>	The context (and thus, planned area of validity) of the analysis

In the context of QualOSS, we have two top-level measurement goals: To measure evolvability and robustness:


<b>Name</b>	Evolvability
<b>Description</b>	Evolvability is the general ability of a F/OSS project to deliver useful products (or product updates) over an extended period of time
<b>Business goals</b>	Productivity, Software Quality
<b>Object</b>	Two objects: (1) the F/OSS product, and (2) the respective community
<b>Purpose</b>	Characterize
<b>Quality focus</b>	Evolvability
<b>Viewpoint</b>	F/OSS users
<b>Context</b>	F/OSS component evaluation

<b>Name</b>	Robustness
<b>Description</b>	Robustness is the general ability of a F/OSS project to deliver robust products over an extended period of time
<b>Business goals</b>	Productivity, Software Quality
<b>Object</b>	Two objects: (1) the F/OSS product, and (2) the respective community
<b>Purpose</b>	Characterize
<b>Quality focus</b>	Robustness
<b>Viewpoint</b>	F/OSS users
<b>Context</b>	F/OSS component evaluation

### 5.3 DEFINITION OF QUALOSS QUALITY INDICATORS

The purpose of this section is to refine the top-level measurement goals into quality characteristics that still representing measurement goals. These quality characteristics are then refined into sub-characteristics, which represent GQM-questions.

Thereby, the quality sub-characteristics are defined such that they represent measurement goals. Using the template presented above, the information on object, purpose, viewpoint, and context will be identical for all measurement goals. Therefore, we will only define quality characteristics themselves, and the information given as part of their definition can be used to complete the measurement goal template in the following way:

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 45 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

<b>Name</b>	--> <i>Name of quality characteristic</i>
<b>Description</b>	--> <i>see description/definition of quality characteristic</i>
<b>Business goals</b>	All goals relate to both business goals: <u>productivity</u> and <u>software quality</u>
<b>Object</b>	Product or community, depending on whether the quality characteristic refers to community or product
<b>Purpose</b>	characterization
<b>Quality focus</b>	--> <i>the name also represents the quality focus</i>
<b>Viewpoint</b>	F/OSS users
<b>Context</b>	F/OSS component evaluation

### 5.3.1 Rationale

In this section, we propose two hierarchies of quality characteristics to define *evolvability* and *robustness*. We consider these hierarchies to be particularly suited for F/OSS projects. The four sources used these hierarchies were the ISO 9126 and IEEE 610 standards, relevant scientific literature, the surveyed F/OSS assessment methodologies, and our interviews with F/OSS users.

To organize our definitions of robustness and evolvability as well as to improve their presentation, we subdivide quality characteristics into subcharacteristics as necessary.

The definition of the quality characteristic hierarchies is based on a few premises:

**Premise 1:** The hierarchies can be tailored to specific situations. Since it is generally easier to perform this customization by discarding irrelevant quality factors rather than by adding new ones from scratch, we have attempted at providing comprehensive hierarchies from which relevant factors can be selected for each case. For example, a system that works in a closed environment may not have to deal with standardization issues since it is not intended to communicate with other software. In that context, quality characteristics related to standardization may simply be removed.

**Premise 2:** Our surveyed scientific literature differentiates software evolvability from software maintainability. while maintainability is an intrinsic characteristic of a product, evolvability also encapsulates the community's ability to handle software changes. It is important to state, that our concept of community encompasses the users, developers and managers involved with the creation of a software product.

### 5.3.2 Evolvability

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. Figure 20 shows the resulting structure.

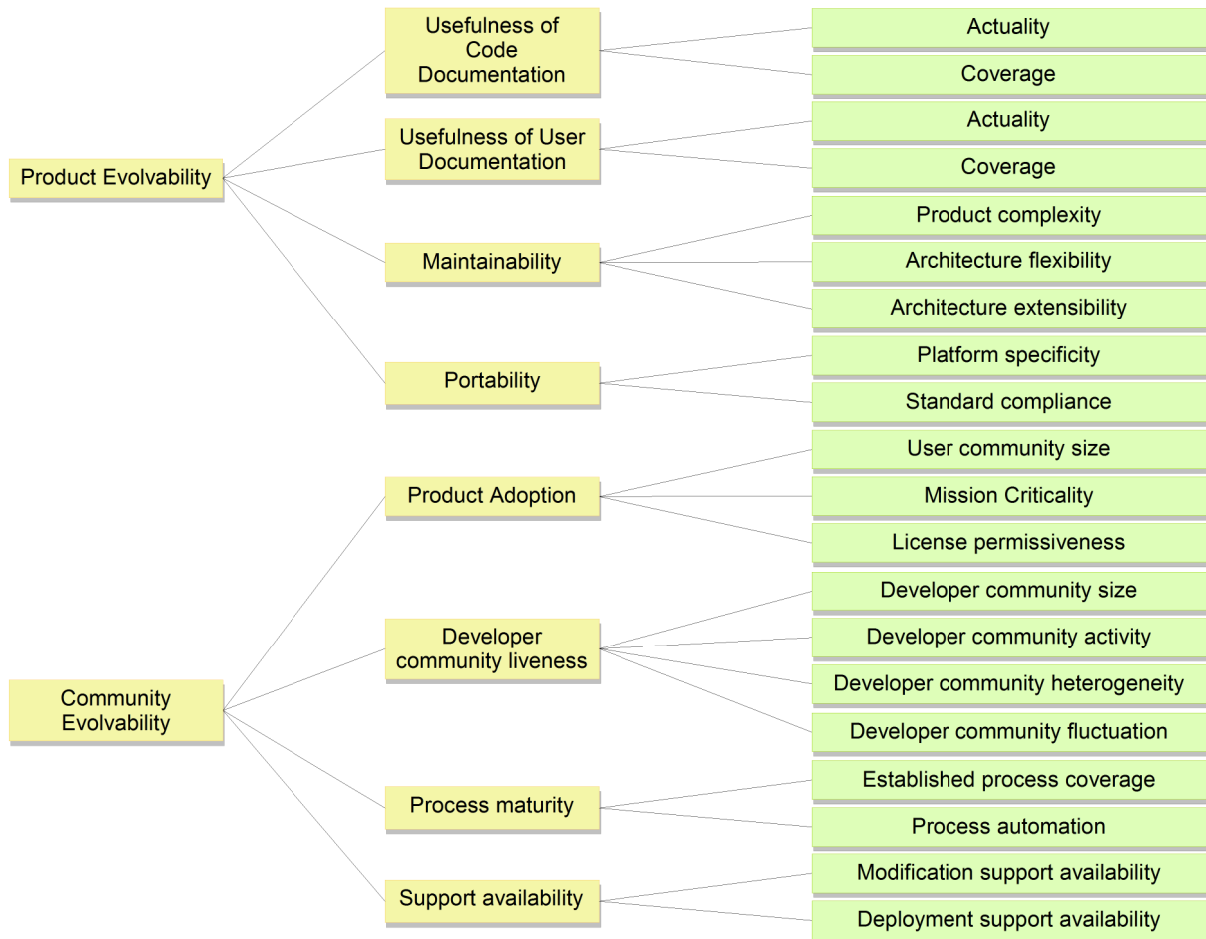




Figure 20: Definition of evolvability

- **Product evolvability:** The ability of a product to be corrected, adapted and extended over time, according to the needs of its users.
  - **Usefulness of code documentation:** The extent to which the source code documentation (documentation explicitly describing the product's internals) is useful when performing corrections, adaptations or extensions to the product.
    - **Actuality:** The extent to which the code documentation describes the current version of the source code as opposite to describing older versions of it.
    - **Coverage:** The ratio between size of documented code and general product code size.
  - **Usefulness of user documentation:** The extent to which the product's user/administrator oriented documentation is useful when deploying and using the product.
    - **Actuality:** The extent to which the user documentation describes the current version of the product functionality as opposite to describing outdated functionality.
    - **Coverage:** The ratio between the number of documented product features and the general number of features offered by the product.
  - **Maintainability:** The 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. IEEE: *The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment.*

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 47 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

- *Product complexity: (IEEE)* The degree to which a system or component has a design or implementation that is difficult to understand and verify.
- *Architecture flexibility:* The ability of the product's architecture of being applied to new problems. *(IEEE)* The ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed.
  - *Extensibility:* The possibility of extending the architecture through external code modules (add-ons, plug-ins) that do not require modifying the program's core. *(IEEE)* The ease with which a system or component can be modified to increase its storage or functional capacity.
- *Portability: (IEEE)* The ease with which a system or component can be transferred from one hardware or software environment to another.
  - *Platform specificity:* The degree to which a product's code is specific to a particular hardware or software environment.
  - *Standard compliance:* The degree to which a product complies with published standards that are relevant to its functionality.
- *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.
  - *Product adoption:* The extent to which a F/OSS product is actively used by individuals and organizations around the world.
    - *User community size:* The number of users (individuals and organizations) that use a F/OSS product worldwide.
    - *Strategic importance: (aka. Mission criticality)* The 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.
    - *License permissiveness:* The amount of freedom allowed to product users by the product's licence.
  - *Developer community liveness:* The amount of work put by a development community into the creation and further development of a software product over a certain period of time.
    - *Developer community size:* The number of individuals and organizations actively contributing to a product's development over a certain period of time.
    - *Developer community activity:* The general number and size of the contributions made to a product's development over a certain period of time.
    - *Developer community heterogeneity:* The degree to which different types of developers (e.g., individuals vs. organizations, for-profit vs. non-for-profit organizations, hobbyists vs. paid professionals) are present in a developer community.
    - *Fluctuation:* The rate movement of people into, and out of a developer community over time
- *Process maturity:* The 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.
  - *Established process coverage:* The 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 (i.e., milestone and roadmap definition, release management), quality assurance (i.e., bug tracking, different forms of code and code change inspections) and requirements engineering (i.e., product improvement proposals.)
  - *Process automation:* The 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.

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 48 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

- *Support availability*: The 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.
  - *Modification support availability*: The availability of support related to performing specific modifications to a software product.
  - *Deployment support*: The availability of support related to solving problems arising from the deployment and use of a software product.

### 5.3.3 Robustness

Similar to evolvability, we define *robustness* at the highest level, as the general ability of a F/OSS project to deliver robust products over an extended period of time (product robustness is defined below). Robustness is influenced by intrinsic product factors as well as by community related factors. Figure 21 shows the resulting structure.

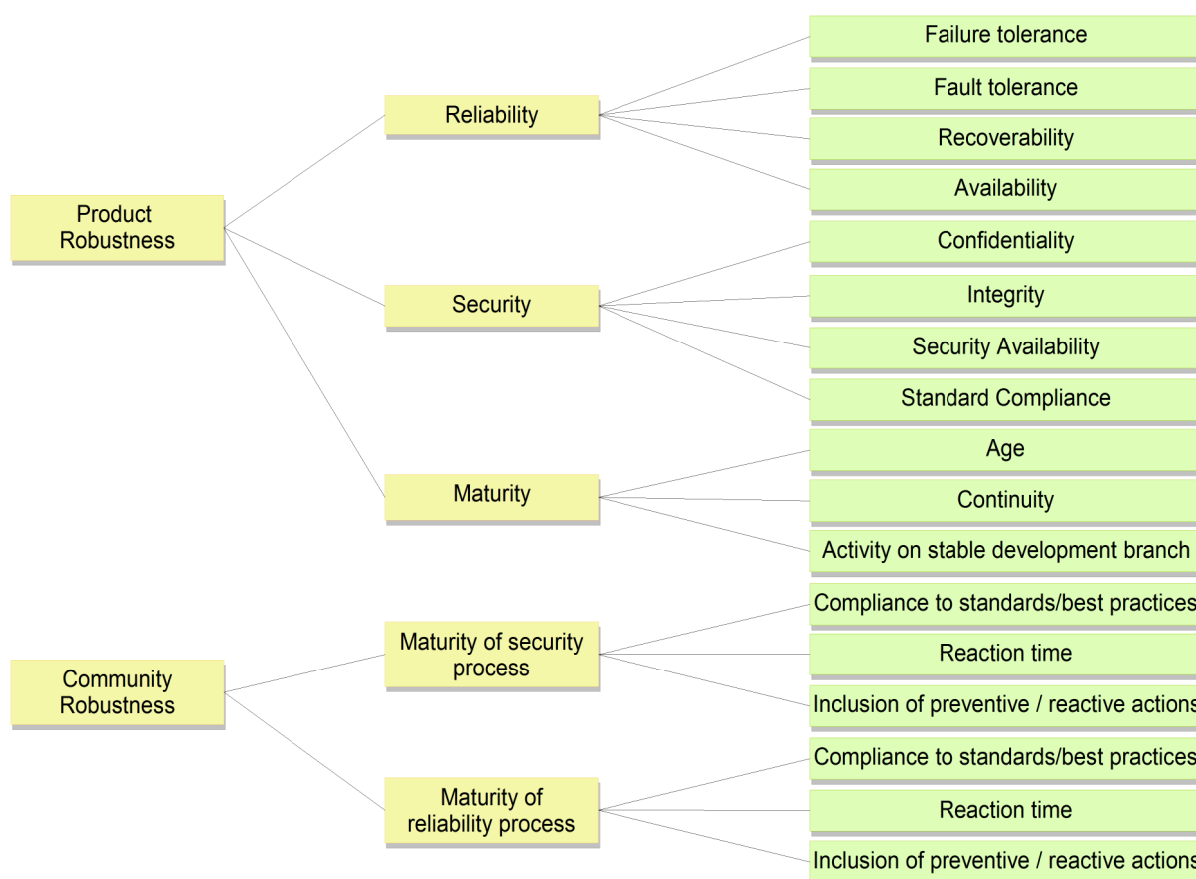




Figure 21: Definition of robustness

- *Product robustness*: (IEEE) The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.
  - *Reliability*: (IEEE) The ability of a system or component to perform its required functions under stated conditions for a specified period of time.
    - *Failure tolerance* (ISO 9126: maturity): The capability of the software product to avoid failure as a result of faults in the software.
    - *Fault tolerance* (ISO 9126): The capability of the software product to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 49 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

- *Recoverability (ISO9126): The 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.*
  - *Availability (IEEE): The degree to which a system or component is operational and accessible when required for use.*
- *Security (ISO 12207): The 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. This includes measures and controls that ensure confidentiality, integrity, and availability of IS assets including hardware, software, firmware, and information being processed, stored, and communicated (CNSS, 2006).*
  - *Confidentiality: The 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. (CNSS, 2006)*
  - *Integrity (ISO): The degree to which a system or component is able to protect the accuracy and completeness of information and processing methods. This includes preventing unauthorised modification or destruction of information (CNSS, 2006).*
  - *Security Availability (ISO): The property of being accessible and useable upon demand by an authorized entity. [ISO 7498-2: 1988]. Or, in other words, timely, reliable access to data and information services for authorized users (CNSS, 2006).*
  - *Compliance to standards: The degree to which a product complies with published standards that are relevant to its functionality.*
- *Maturity: The degree to which the general, long-term objectives set for a product have been reached by the current implementation.*
  - *Age: The time span over which a product has been developed.*
  - *Continuity: The regularity with which community contributions have been made to the a product over its lifespan.*
  - *Activity on stable development branch: The number and size of the contributions made to a product's stable development branch over a certain period of time. High activity on a branch declared to be stable can be a sign of low product maturity.*
- *Community robustness: The ability of the established processes in a community to guarantee the delivery of robust products.*
  - *Maturity of security process: The degree to which a development community has established processes dedicated to guarantee the security of delivered products. Also, the degree to which a community reacts effectively and timely when a security defect is found in a released product.*
    - *Compliance: The degree to which the processes and procedures dealing with security adhere to best practices and security standards*
    - *Reaction time: The amount of time that is typically required for resolving security-related issues*
    - *Inclusion of preventive/reactive actions: The degree to which the community commits to actions aimed at preventing security problems*
  - *Maturity of reliability process: The 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). Also, the degree to which a community reacts effectively and timely when a critical defect is found in a released product.*
    - *Compliance: The degree to which the processes and procedures dealing with reliability adhere to best practices and security standards*
    - *Reaction time: The amount of time that is typically required for resolving reliability-related issues*
    - *Inclusion of preventive/reactive actions: The degree to which the community commits to actions aimed at preventing reliability problems*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 50 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

### 5.3.4 Quality Factors Not Considered by QualOSS

There are several quality factors that are important to F/OSS users that will not be considered by the final QualOSS model:

- *Usability*: The degree to which a product can be easily used, deployed, and administrated is a major factor for adoption of a product. QualOSS does not focus on evaluating usability, however, although usability and ease of deployment are partially evaluated as sub-characteristics to support availability.
- *Functionality*: The degree to which the F/OSS component fulfils desired functionality is a main criterion for the decision on whether to adopt it. However, this always depends on the concrete user's needs and situation, and a generic model for functionality can thus not be defined.
- *Legal/License issues*: Another major concern for adoption of F/OSS components is whether the intended usage conflicts with the F/OSS license. However, neither robustness nor evolvability are directly or indirectly influenced by these issues.

## 6 VALIDATION PLAN

This section presents the initial plan for validation of the QualOSS model to be developed in task 1.3. The initial plan states evaluation goals and main constructs to be evaluated. The refinement of the validation plan in terms of definition of hypotheses, metrics, and a study design, will be done as part of tasks 1.4-1.6. In particular, the validation plan developed in workpackage 1 will form a validation framework that will be enhanced and further refined in workpackages 4 and 5.

Validation goals have to be derived from goals of the technology or approach they are to evaluate. In the case of QualOSS, the goals can be stated as follows:

- (A) The QualOSS model should deliver useful and reliable information to stakeholders
- (B) The QualOSS model reduces the risks involved in using F/OSS components
- (C) The QualOSS model reduces the time for assessment of F/OSS component
- (D) The QualOSS model helps OSS communities improve

This leads to two main categories of evaluation goals:

- Evaluate the appropriateness of the QualOSS model for the stakeholders (technology goals A and B)
- Evaluate the impact of the QualOSS model (technology goals C and D)


In the following, detailed evaluation goals for these categories will be listed and described using the GQM goal template.

### 6.1 GOAL CATEGORY 1: EVALUATION OF APPROPRIATENESS OF QUALOSS MODEL

The evaluation goals in this category are concerned with evaluating whether the QualOSS model delivers useful and reliable information to the stakeholders; that is, to evaluators of F/OSS components. The following list defines some potential evaluation goals:

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

<b>Name</b>	<b>EG1:</b> Validity of the QualOSS model's definition.
<b>Description</b>	Evaluate definition of quality model with stakeholders.
<b>Object</b>	QualOSS model definition
<b>Purpose</b>	characterize
<b>Quality focus</b>	Validity of the QualOSS model (i.e., of the quality characteristic definition and prioritization) compared to perception and intuition of QualOSS evaluators

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 51 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

<b>Viewpoint</b>	F/OSS evaluator
<b>Context</b>	QualOSS evaluation in tasks 1.5 and 1.6
<b>Evaluation Plan</b>	This can be done, for example, through interviews or a survey. The concrete setting will be decided in task 1.6

**Evaluation goal 2:** Evaluate usefulness and usability of QualOSS model.


<b>Name</b>	<b>EG2:</b> QualOSS model usefulness.
<b>Description</b>	Evaluate usefulness and usability / ease of use of quality model for stakeholders.
<b>Object</b>	QualOSS model
<b>Purpose</b>	characterize
<b>Quality focus</b>	<b>Usefulness;</b> i.e., the degree to which a person believes that the QualOSS model provides support for effective evaluation of F/OSS components. <b>Usability;</b> that is, the degree to which a person believes that using a particular system would be free of effort. For the QualOSS model, this includes <ul style="list-style-type: none"> <li>• Ease of use of the QualOSS tools</li> <li>• Interpretability of the QualOSS model output.</li> </ul>
<b>Viewpoint</b>	F/OSS user / evaluator
<b>Context</b>	QualOSS evaluation in tasks 1.5 and 1.6
<b>Evaluation Plan</b>	This can be done, for example, through a questionnaire to be filled in by users of the QualOSS model. The concrete setting will be decided in task 1.6

**Evaluation goal 3:** Evaluate 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.

<b>Name</b>	<b>EG3:</b> Validity and reliability of the QualOSS model.
<b>Description</b>	Evaluate validity and reliability of the quality model for stakeholders.
<b>Object</b>	QualOSS model
<b>Purpose</b>	characterize
<b>Quality focus</b>	<b>Validity:</b> The degree to which a set of measurements measure the intended construct. A valid measure is one which is measuring what it is supposed to measure. Validity implies reliability (consistency) <b>Reliability:</b> The degree to which a set of measurements is consistent; in other words: the extent to which a set of questions can be treated as measuring a single construct (or latent variable, such as robustness). In this case, whether the metrics proposed for the quality (sub-)characteristics are consistently measuring the same characteristic.
<b>Viewpoint</b>	F/OSS user / evaluator
<b>Context</b>	QualOSS evaluation in tasks 1.5 and 1.6
<b>Evaluation Plan</b>	This can be done, for example, by applying the QualOSS model to a set of F/OSS projects. Reliability can be calculated by comparing the values of metrics for different characteristics. Validity needs also to take into account alternatives for measuring the "true" quality characteristic; for example, by contrasting the measurement with stakeholders' perceptions. The concrete setting will be decided in task 1.6

## 6.2 GOAL CATEGORY 2: EVALUATION OF QUALOSS MODEL IMPACT:

The evaluation goals in this category are concerned with evaluating whether the quantifiable benefit that the QualOSS model creates for stakeholders; that is, for evaluators of F/OSS components. As this implies that the QualOSS model is applied in real projects, evaluation of these aspects is part of workpackage 5. The following list defines some potential evaluation goals:

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 52 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

**Evaluation goal 4:** Evaluate whether using the QualOSS model reduces the risk involved in using F/OSS components from stakeholders' viewpoints

<b>Name</b>	<b>EG4:</b> Risk reduction through QualOSS model .
<b>Description</b>	Evaluate the degree to which the QualOSS quality model reduces the risk of using F/OSS components for stakeholders.
<b>Object</b>	QualOSS model
<b>Purpose</b>	characterize
<b>Quality focus</b>	Degree of risk reduction
<b>Viewpoint</b>	F/OSS user / evaluator
<b>Context</b>	QualOSS evaluation in workpackage 5
<b>Evaluation Plan</b>	This can be done, for example, through a questionnaire (aimed at <i>perceived risk reduction</i> ) to be filled in by users of the QualOSS model. The concrete setting will be decided in workpackage 5

**Evaluation goal 5:** Evaluate whether using the QualOSS model reduces the cost for F/OSS assessment


<b>Name</b>	<b>EG5:</b> Assessment cost reduction through QualOSS model .
<b>Description</b>	Evaluate the degree to which the QualOSS quality model reduces the cost of assessing F/OSS components.
<b>Object</b>	QualOSS model
<b>Purpose</b>	characterize
<b>Quality focus</b>	Degree of assessment cost reduction. Assessment cost can be measured through effort required for F/OSS assessment
<b>Viewpoint</b>	F/OSS user / evaluator
<b>Context</b>	QualOSS evaluation in workpackage 5
<b>Evaluation Plan</b>	This can be done, for example, in a case study setting, by measuring the amount of effort required by manual assessment, and by contrasting these findings to those of the QualOSS model. The concrete setting will be decided in workpackage 5

**Evaluation goal 6:** Evaluate whether the OSS communities accept and adopt the QualOSS model. It is one of the goals of QualOSS to create such an impact in the F/OSS community. However, it is questionable whether this impact can be expected until the end of the project.

<b>Name</b>	<b>EG6:</b> Adoption of QualOSS model .
<b>Description</b>	Evaluate the degree to which the QualOSS quality model is adopted by the F/OSS community.
<b>Object</b>	QualOSS model
<b>Purpose</b>	characterize
<b>Quality focus</b>	Degree of adoption
<b>Viewpoint</b>	F/OSS user / evaluator
<b>Context</b>	QualOSS evaluation in workpackage 5
<b>Evaluation Plan</b>	This can be done, for example, by determining the number of F/OSS projects that adopt the QualOSS model. The concrete setting will be decided in workpackage 5

## 7 SUMMARY AND CONCLUSIONS


This report defines the goals and measurement requirements for QualOSS quality models. In this report, we review relevant definitions of robustness and evolvability in F/OSS assessment approaches and in the state of the art and practice of quality models. Additionally, we take into account stakeholders' perceptions and requirements through a series of interviews.

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 53 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

Goals and requirements for the QualOSS model are defined in terms of (a) business and measurement goals, (b) a consolidated definition of quality characteristics for evolvability and robustness from related work and from stakeholders' views, and (c) an initial plan for validation of the QualOSS model.


Further work is still required. In particular, the QualOSS model needs to be further refined into metrics; this is part of task 1.3. We foresee that part of task 1.3 will be to develop an assessment method for evaluating the community maturity, as approaches to evaluate associated processes have so far not been considered in F/OSS assessment methods.

In addition, further refinement of the initial validation plan will be done in tasks 1.4-1.6. Main goals of the validation in workpackage 1 will be to evaluate the validity of the QualOSS model's definition with stakeholders. The workpackages 4 and 5 will continue to refine the validation plan. Workpackage 5 will particularly be concerned with validating business goals and the validity and reliability of the QualOSS model.


	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 54 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

## REFERENCES

- (Agresti et al., 2002) William W. Agresti, Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach, Encyclopedia of Software Engineering. Volume 1., ch. Measurement, pp. 762--775. John Wiley Sons, 2002
- (Balzert, 1998) Helmut Balzert, Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Spektrum, Aka-demischer Verlag, 1998
- (Basili, 1985) V. R. Basili, "Quantitative Evaluation of Software Engineering Methodology", Proceedings of the First Pan Pacific Computer Conference, Melbourne, Australia, Vol. 1 pp. 379-398, 1985.
- (Basili, 1992) Basili, Victor R., "Software Modeling and Measurement. The Goal/Question/Metric Paradigm", Computer Science Technical Report Series NR: CS-TR-2956 / NR: UMIACS-TR-92-96, 1992.
- (Basili et al., 1994) Basili V.R., Caldiera G., and Rombach H.D. Experience Factory, The Encyclopedia of Software Engineering, Volume I, Pages 469-476, John Wiley & Sons, 1994.
- (Basili et al., 2002) V. R. Basili, F. E. McGarry, R. Pajerski, and M. V. Zelkowitz. Lessons learned from 25 years of process improvement: The rise and fall of the NASA software engineering laboratory. In IEEE Computer Society and ACM International Conference on Software Engineering (ICSE 2002), May 2002
- (Basili and Rombach, 1988) V.R. Basili, H.D. Rombach, "The TAME Project: toward improvement-oriented software environments", IEEE Transactions on Software Engineering, 14:6, p.759-773, 1988.
- (Basili and Weiss, 1984) Basili V.R. and Weiss D. M. A methodology for Collecting Valid Software Engineering Data, IEEE Transactions on Software Engineering, 10(6):728-738, Nov. 1984.
- (Basili et al., 1995) V. Basili, M. Zelkowitz, F. McGarry J. Page, S. Waligora, and R. Pajerski. SEL's software process improvement program. IEEE Softw., 12(6):83-87, 1995.
- (Boehm et al., 1973) B.W. Boehm, J.R. Brown, H. Kaspar, M. Lipow, G.J. MacLeod, and M.J. Merritt. Characteristics of Software Quality. TRW Software Series TRW-SS-73-09. December, 1973.
- (Boehm et al., 1976) B. W. Boehm, J.R. Brown, and M. Lipow. Quantitative evaluation of software quality. In Proceedings of the 2nd International Conference on Software Engineering (ICSE) 1976, pages 592-605, 1976.
- (Boehm et al., 1978) Barry W. Boehm, John R. Brown, Hans Kaspar, Myron Lipow, Gordon J. MacLeod, and Michael J. Merritt, Characteristics of Software Quality. North Holland Publishing Company, 1978.
- (Bøegh et al., 1999) Jørgen Bøegh, Stefano Depanfilis, Barbara Kitchenham, and Alberto Pasquini. A method for software quality planning, control, and evaluation, IEEE Software, March/April:69-77, 1999.
- (Briand et al., 1997) Lionel C. Briand, Christiane Differding, and H. Dieter Rombach, Practical Guidelines for Measurement-Based Process Improvement, Software Process Improvement and Practice Journal, vol. 2, no. 3, 1997.
- (McCall et al., 1977) J.A.McCall, P.K.Richards, G.F.Walters, "Factors in Software Quality", RADC TR-77-369, 1977. Vols I,II,III", US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055, 1977.

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 55 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted


- (DGQ, 1986) Deutsche Gesellschaft für Qualität, Software-Qualitätssicherung. No. DGQ-NTG-Schrift 12-51, VDE-Verlag Berlin, 1986.
- (Duijnhouwer and Widdows, 2003) F. Duijnhouwer, C. Widdows: Open Source Maturity Model, Capgemini Expert Letter, August 2003
- (Freimut et al., 2003) Freimut, Bernd; Assmann, Danilo; Kaiser, Peter; Trendowicz, Adam; Wieczorek, Isabella: Quality Models to Manage Software Development - A State of the Art Report, IESE-Report, 035.03/E , 2003
- (Fenton and Pfleeger, 1996) Norman E. Fenton and Shari Lawrence Pfleeger, Software Metrics - A Practical and Rigorous Approach. International Thomson Computer Press, 2nd edition ed., 1996
- (Golden, 2005) B. Golden: Making Open Source Ready for the Enterprise: The Open Source Maturity Model, Navica White Paper, [www.navicasoft.com](http://www.navicasoft.com), extracted from B. Golden: "Succeeding with Open Source", Addison-Wesley, 2005
- (Grady and Caswell, 1987) Robert B. Grady and Deborah L. Caswell, Software Metrics: Establishing a Company-Wide Program, Prentice-Hall, 1987
- (Grady, 1992) Robert B. Grady, Practical Software Metrics for Project Management and Process Improvement. Prentice Hall, 1992
- (Gresse et al., 1995) Gresse B., Hoisl B., Wüst J. A Process Model for GQM-Based Measurement, STTI-95-04-E, Department of Computer Science, University of Kaiserslautern, Germany, 1995.
- (Hartkopf et al., 2007) Susanne Hartkopf, Christian Denger, Ronny Kolb, Adam Trendowicz: State of the Art in Quality Modeling for Reuse-Based Software Development in Small and Medium Sized Enterprises, LifeCycleQM Report, 2007
- (Hyatt and Rosenberg, 1996) Larry Hyatt and Linda Rosenberg, A software quality model and metrics for risk assessment, in Proceedings of the ESA 1996 Product Assurance Symposium and Software Product ASS, 1996.
- (ISO 8402) ISO 8402, Quality management and Quality assurance Vocabulary
- (ISO 9126) ISO/IEC 9126 International Standard, Software engineering – Product quality, Part 1: Quality model, 2001.
- (ISO 12207) ISO/IEC 12207: International Standard, Information technology -- Software life cycle processes, 1995
- (ISO 14598) ISO/IEC 14598 International Standard, Standard for Information technology -- Software product evaluation -- Part 1: General overview
- (Kitchenham et al., 1997) Barbara Kitchenham, Steve Linkman, Alberto Pasquini, and Vincenzo Nanni, The SQUID-Approach to Defining a Quality Model, Software Quality Journal, vol. 6, no. 3, pp. 211-233, 1997.
- (CNSS, 2006) Committee on National Security Systems (CNSS) Secretariat, National Security Agency: National Information Assurance Glossary, CNSS Instruction No. 4009, June 2006
- (OpenBRR, 2005) OpenBRR.org, Business Readiness Rating for Open Source: A Proposed Open Standard to Facilitate Assessment and Adoption of Open Source Software , BRR 2005. <http://www.openbrr.org>

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 56 of 88</p>
		<p>Version: 1.3 Date: Mar 15, 07</p>
		<p>Status : Proposal Confid : Restricted</p>

(Park et al., 1996) Park, R, W. Goethert, W. Florac, 1996. Goal-driven software measurement – a guidebook, SEI report CMU/SEI-96-HB-002.

(QSOS, 2006) QSOS Project, Method for Qualification and Selection of Open Source software (QSOS), version 1.6, April 2006. <http://www.qsos.org>

(van Solingen and Berghout, 1999) R. Van Solingen, E. Berghout, The Goal/Question/Metric Method, A Practical Guide for Quality Improvement of Software Development, Mc Graw Hill, 1999

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 57 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

## APPENDIX A: USAGE SCENARIO INTERVIEW SHEET

### INSTRUCTIONS

The goal of the interviews is to elicit usage scenarios for F/OSS components and associated evaluation criteria. The results will be used to define goals for the QualOSS quality models.

The interview is mainly directed at organizations that are currently using F/OSS in any way. Candidates are organizations using F/OSS as part of their information infrastructure, and organizations that are offering F/OSS based products or services to the public, among others. Additionally, the interview can be also conducted with organizations not currently using F/OSS, in order to determine their reasons not to use it.

Use the questions below as a base for conducting the interview. All questions are followed by an explanation (in a box) that is intended to guide the interviewer. If the interviewee does not understand a term, the interviewer can provide guidance by using some of the suggested hints and additional questions. Following the explanation, there is also a sample answer (in *cursive*) which illustrates how the interviewee's answers could be documented. Parenthesis are used to indicate variants of the questions that could be used with organizations that are *not* using F/OSS.

This sheet is intended to be used by QualOSS members in in-house interviews.

### ORGANIZATIONAL INFORMATION

Date of Interview	
Start Time	
Company	Removed from the appendix for anonymization
Context / Domain	
Interviewee Name	Removed from the appendix for anonymization
Interviewee Position	Removed from the appendix for anonymization
Interviewee contact (e.g., Email)	Removed from the appendix for anonymization


### INTERVIEW INTRODUCTION

Explain to interviewee: F/OSS components are growing in importance today. One critical question in selecting F/OSS components is to find out whether they are good enough to be used in their goal context. We are conducting this interview to find out how F/OSS components are selected and used in your company, or, alternatively, why your company is not using them now.

### INTERVIEW QUESTIONS

#### Q1: For which types of purposes/systems do you (would you) use F/OSS products?

Example possible answers: server, desktop, (Internet) infrastructure, part of a commercial product, part of an in-house application, etc.

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 58 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

It is important that the interviewee can come up with his/her own classification. This classification will be used in all following questions to differentiate answers. That is, for all following questions, ask for differences between types.

When documenting the interview results, use a separate document for each type. For example, if they use (or would use) F/OSS for server and desktop computers, create one document that summarizes the answers for server, and another one that summarizes the answers for desktop computers

*We have a public website (<http://www.wagse.informatik.uni-kl.de>) which is used to provide general information about our group (mainly to students) and to hold special web pages for the lectures and seminars we offer. A small intranet is also available for sharing information between our group's members and storing internal documents.*

*Both of these are based on OSS infrastructure.*

## **Q2: Which F/OSS products do you use? (Have you considered any concrete products you would use?)**

In case too many different products are used, the interviewee try to find at least some examples for each category.

For example, if the interviewee stated that they use F/OSS products for server and for desktop purposes, the question here can be something like: "You stated that you use F/OSS products for server and for desktop purposes. Can you give examples of products you use on the server? Can you give examples of products you use on desktop computers?" This is also applicable to products the interviewee's organization has only considered but do not use yet.

Try to cover all categories mentioned by the interviewee for the following questions, too.

**If the interviewee's organization is not currently using F/OSS software, skip to question 6.**

*Zope and a number of Zope products (add-ons).*

## **Q3: How do you use F/OSS products?**


Find out in which ways F/OSS products are being used in the interviewee's organization.

This question can contain many other subquestions: Do you make modifications to the F/OSS components? Do you integrate them by writing glue code? Do you use a F/OSS library? Do you port F/OSS component to a new platform? ...

*We use Zope as our main web server, extended with a number of OSS extension packages. We also wrote a number of scripts that do particular tasks, like creating a news bar at the main page, and tabulating the contact information for our group's members. Additionally, we created a Zope product that we use to support the workflow in some of our courses.*

## **Q4: Why did you choose these F/OSS components?**

Determine why this product was chosen against other possible F/OSS or commercial offers.

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 59 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

*We needed a flexible system that would not require us to centralize the administration of our web server. This is possible with Zope. For example, at this time, when a group member is responsible for a lecture, it is possible for him to directly create and update the corresponding web pages without the risk of disrupting the rest of the server.*

*It was also convenient that Zope was freely available and had a large number of available extensions. The fact that I had previous experience with Zope also played a role in the decision.*

### **Q5: To your knowledge, do you achieve business goals through F/OSS usage?**

Determine what the interviewee and/or his organization want to achieve by using F/OSS components, and whether this is directly aligned with any organizational business goals. In a broad sense, we understand *business goals* as those related to the success and profitability of an organization.

*Since we belong to a public institution, we cannot strictly speak of business goals in our case. On the other hand, it is our goal to present our students with a good academic offer, and a well-organized, up-to-date web site is part of that. Arguably, our students are now better informed about our academic offer. We can also better support our lectures and seminars by making it easy for teachers and assistant to quickly publish information.*

### **Q6: What are criteria that F/OSS components need to fulfill to be appropriate?**

Identify the interviewee's quality expectations toward F/OSS components. Example possible answers: high quality product, low development effort, access to source code and possibility to implement required functionality, ...

*When we started with Zope we expected it to be reliable and flexible, and it has fulfilled both expectations good enough for our purposes. We also had expectations regarding support. Since we didn't have a large budget for the web site, it was important to have a minimum of support from the Internet (mailing lists, fora, etc). This has worked well up to now. Actually, you find most solutions by searching the Internet. Directly asking in mailing lists, for example, hasn't been necessary.*


### **Q7: How did you (would you) evaluate these criteria?**

The question before asks what the interviewee wants to know about F/OSS components. This question addresses what they do to find out whether the component satisfies their criteria.

*The only way we had to evaluate these criteria was by looking at what other people had to say about the product. Searching the Internet quickly showed us a large number of apparently satisfied users. Looking at Zope's web pages and mailing list were also signs of an active community.*

### **Q8: Did you (would you) consider any criteria related to Robustness and Evolvability?**

We should explicitly ask for these, as they are in the focus of QualOSS. If some robustness/evolvability criteria have been mentioned before, these can be used to help to guide the rest of the interview.

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 60 of 88</p> <hr/> <p>Version: 1.3 Date: Mar 15, 07</p> <hr/> <p>Status : Proposal Confid : Restricted</p>
---	--	--

If the interviewee's organization is not currently using F/OSS software, skip to question 11.

### Q9: How did you evaluate this criterion?

Repeat this question for each criterion mentioned by the interviewee.

*Usability: We looked at mailing lists to determine user satisfaction*

*Support: Look at activity in mailing lists*

### Q10: If not evaluated: Why did you not evaluate the criterion?

Reliability: We did not know how to test the product for reliability


### Q11: How important is this criterion to you?

Rate on the following scale of 0 to 10, where 0 means that the criterion is irrelevant, and 10 means that a high quality level with respect to this criterion is critical for selecting the F/OSS component.

*Usability: 8 (We have to give the product to untrained users, usability is fairly critical.)*

*Support: 10 (If we have questions, they need to be answered really quickly, because this usually means that our sales process could be stopped until the question is resolved)*

*Reliability: 7 (...)*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 61 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

## APPENDIX B: INTERVIEW RESULTS

### INTERVIEW 1

#### Organizational Information

Date of Interview	07/11/06
Interviewer	MERIT
Start Time	12:00
Context / Domain	All services around open source. From developers to advice, support, training, network services.

#### Interview Questions

##### Q1: For which types of purposes/systems do you use F/OSS products?

Time:03:11

*We try to use open source as much as we can when we find provisional solutions. We use it for almost all the communication solutions except application which are "too specific", for example accounting, some for production units. For these applications it is difficult, but for the rest on mailing, on security, on file server, on lot of domains of inter-enterprise communication.*

*The other part where we go slowly is the user desktops. We make some migration for this, and we use desktop but it is in particular cases when we know it will work.*

##### Q2: Which F/OSS products do you use?

Time: 05:00


*For mail we use Cyrus IMAP, Postfix. We use a lot of webmail with like IMP Horde, we use the network part of Linux for security for e.g. IP-Tables, we use Samba for the file server and printing server, we use Sympa for mailing lists. We use a lot Apache with PHP, and MySQL and PostgreSQL. Tomcat for Java applications.*

##### Q3: How do you use F/OSS products?

##### Q4: Why did you choose these F/OSS components?

Time: 06:38

*Some of them are at level of proprietary solutions, sometimes they are better. For the high load, they work sometimes better. In our case, we will specialize in open source, so we sometimes we do not know enough proprietary solutions, so we are more efficient with open source.*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 62 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

### **Q5: To your knowledge, do you achieve business goals through OSS usage?**

*Time: 07:35*

*Our goal is to propose services around open source for our clients. We are a company in computing services, so it follows naturally that open source helps us reach the business goals.*

### **Q6: What are criteria that F/OSS components need to fulfill to be appropriate?**

*Time: 08:23*

*Reliability, performance and usability.*

*We choose solutions which are the most commonly used, because sometimes we have a choice, and for our clients we do not use a solution that is too young. In our case we can try things, but for our clients we need to give them reliable solutions.*

### **Q7: How did you evaluate these criteria?**

*Time: 09:18*

*--what do you mean with "too young" exactly and what is "reliable" exactly?*

*That is difficult. We have been in the domain and area for 10 years, and we have experts in our company, and we know on which solution we can rely on. We try to grade a community/evaluate it w.r.t. If it is dynamic, we need a number of projects known in big companies, we spend a lot of time to check this criteria. In fact we have the same results in consulting our experts without having to do a lot of research on it.*

### **Q8: Did you consider any criteria related to Robustness and Evolvability?**

*Time: 10:50*

*It is important, how they evaluate them.*

*We control if community is dynamic and to see if it will go on further, and if the solution will be maintained over time.*

*-->what is dynamic and how do you decide?*


*You have to see how big is the community, with firms in it. How fast they respond to bugs and questions on the forums, which frequency new versions are released with new bug fixes and new versions.*

*I think for evolvability and robustness it is quite the same.*

### **Q9: How did you evaluate this criterion?**

--

### **Q10: If not evaluated: Why did you not evaluate the criterion?**

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 63 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

## Q11: How important is this criterion to you?

Time: 12:34

*It depends on the clients. Sometimes we need all points.*

*--> We followed ISO/IEC 9126/1 list as URJC did, to grade this and give reason to get some more real answers.*

*Maturity: 7 – We have to be sure the solution will be sufficiently tested and used, and that it will not change into something totally different in next release.*

*Fault tolerance: 5 – For our clients is not each time important. Sometimes we have to build something around one solution to make it fault tolerant with other OSS components, but sometimes the solution itself is not fault tolerant, but we make it tolerant.*

*Maintainability: 7 – usually in OSS solutions it is often documented for big projects, it is easier to maintain. It is important, but almost all solutions we use have this criteria.*

*Analyzability: ? 5 - To maintain we have to know the code, and we have to analyze it to maintain, and to check security, to check features. But only 5 because we when we know the solution well, and we know what we can do and where it can go.*

*Changeability: You mean if it is to change to another solution? 4 – it often it is not question of our client. We think these solutions are more, .. the risk is less important with open source solutions than proprietary solutions that something like apache needs to change because it has been open up. And for changing things ourselves – 7 in importance to configure and meet requirements of clients. We rarely use out of the box solutions.*

*Stability: 10 – very important, because when the client chose OSS instead of proprietary solution we have to do at least as well on stability as proprietary solution. So we propose support for client every day to make sure it is stable. Very important factor.*

*Testability: Every application must be tested, but often of course you can test it and check if it works. 7 – but we can do it.*


*Adaptability: It really depends on solution. Sometimes we need to do big adaptations it is very important, for others (e.g. Fileserver), we do not have to do so much so it is less important. Other times it is so complicated or big application like samba we do not adapt it so much. But maybe for e.g. Squid proxy we have to change some features, so in this case it must be easy to use modules or to be able to change it. So it depends. It can go from 5-9.*

*Installability: Easy to install? We know that for some solution we have to spend a lot of time to install configuration, and it is not a problem because for it is a very complicated or specific use. So we know some products cannot be used out of box directly, because it is complicated, and we need to communicate with windows systems and others. So, it has to be documented to install it, but all of them have it to be used. But, from 5-8. But, it is not the principal part for us.*

*Coexistence: ? What does it mean? Really important, almost all OSS can coexist on same server, it is a 9, but almost all of them do.*

*Replacability: Easy to replace with another? Not important, a 4.*

*Safety: In terms of security? Very important. Sometimes it can be a bit different if solution is on Internet or inside company, but in all cases it a 8-10.*

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 64 of 88</p>
		<p>Version: 1.3 Date: Mar 15, 07</p>
		<p>Status : Proposal Confid : Restricted</p>

## INTERVIEW 2

### Organizational Information

Date of Interview	25 - october -2006
Interviewer	URJC
Start Time	10.00 am
Context / Domain	Open Source Integrator

### Interview Questions

**Q1: For which types of purposes/systems do you (would you) use F/OSS products?**

*Merging functionalities in order to get complex capabilities*

**Q2: Which F/OSS products do you use? (Have you considered any concrete products you would use?)**

*We have not considered the use of any concrete product. we use the "bests ones"*

**Q3: How do you use F/OSS products?**

*Usually, implementing and integration software layer to make the foss to cooperate with others.*

**Q4: Why did you choose these F/OSS components?**

*We can access the source and so we can modify and enhance it for our enterprise purposes*

**Q5: To your knowledge, do you achieve business goals through F/OSS usage?**


*Yes. This is the main business area of our company. We define new products using several of FOSS ones.*

**Q6: What are criteria that F/OSS components need to fulfill to be appropriate?**

*Mainly: functionality, suitability, no beta versions, standards based, reliability, maturity and that it is not a closed project.*

**Q7: How did you (would you) evaluate these criteria?**

*1st the public info is evaluated, then several "unit" tests will make so we can test the functionality as well as the reliability.*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 65 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

**Q8: Did you (would you) consider any criteria related to Robustness and Evolvability?**

*Yes, we look for no closed projects*

**Q9: How did you evaluate this criterion?**

*Reading the Roadmap (we believe it), previous timeline and for robustness usually code inspections*

**Q10: If not evaluated: Why did you not evaluate the criterion?**

**Q11: How important is this criterion to you?**

*functionality: 8, reliability: 9, standardized: 8*

**INTERVIEW 3**

**Organizational Information**

Date of Interview	October, 24th 2006
Interviewer	FUNDP
Start Time	2.40 AM
Context / Domain	IT enterprise. It promotes the use of IT in public sector

**Interview Questions**

**Q1: For which types of purposes/systems do you (would you) use F/OSS products?**

*Answer:*


*We use OSS infrastructure in many cases:*

- Servers in operating system like Linux
- Applications servers like APACHE, TOMCAT, JBOSS
- Tools for modelling
- Tools Integration framework to build web applications
- Web Content Management Systems

**Q2: Which F/OSS products do you use? (Have you considered any concrete products you would use?)**

*Answer:*

*We use OSS products like Linux, GBOSS, TOMCAT, ECLIPSE, J2EE Container and persistence libraries.*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 66 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

### Q3: How do you use F/OSS products?

Answer:

*Most of time, we use OSS libraries. But when the OSS is well documented, legible and readable, we can use it without modifications . Sometimes, we also write a number of scripts and add it to OSS to have more functionalities. There are also many OSS without documentation and it is difficult to know the real behavior of these OSS.*

### Q4: Why did you choose these F/OSS components?

Answer:

*Generally, we have to follow our customers requirements and IT standards that includes the use of OSS. They ask us to use OSS components whenever it is possible. In others cases, it is our initiative particularly when we want:*

- To reduce the cost of building applications
- To reuse code of OSS
- To be flexible and not to be attach to a particular constructor
- to benefit of a large community of users for support

*Sometimes, it is because of the popularity of the OSS. Finally, OSS are freely available and we are familiar with some of them.*

### Q5: To your knowledge, do you achieve business goals through F/OSS usage?

Answer:

*It depends. We don't use OSS for databases, workflow systems or Business Intelligence systems because they have not yet reach the minimal requirements level. Apart from that we do achieve our business goals with OSS.*

### Q6: What are criteria that F/OSS components need to fulfill to be appropriate?


Answer:

*To be appropriate, an OSS components need to be :*

- Mature
- Popular
- Known as belonging to a group with some specific standardization criteria (for example Jakarta, Apache, ....)
- In case we want to modify code, the OSS components need to be legible and readable, well documented, well structured and to have a strong and coherent architecture.

### Q7: How did you (would you) evaluate these criteria?

Answer:

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 67 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

*If possible, we can evaluate it by using some scientific methods (metrics, ...) but it is costly. We evaluate them in a specific project or have some information from Internet. It could be:*

- The maturity : for how many long the project exists?
- The support (documentation, architecture, ...)
- The team working on the project: there are known people in OSS community who have good reputation.
- The views of the OSS community on the OSS
- Users opinions on these OSS (Forum, ...).

### **Q8: Did you (would you) consider any criteria related to Robustness and Evolvability?**

*Answer:*

*For Robustness, some criteria could be:*

- Maturity (life cycle, tests' results, ...);
- The way the product is realized ( with standard norms, ...);
- Well documented (Wiki, Forum, ...).

*For evolvability, some criteria could be:*

- The use of recognize standard to realize the product;
- Application should resolve a general problem and not to be too specific;
- Well documented (Wiki, Forum, ...)

### **Q9: How did you evaluate this criterion?**

*Answer:*

*We can evaluate it when it is possible with scientific methods but in general we obtain informations from the net.*

- The maturity : for how many long the project exists?, the opinions of users of this OSS (Forum, ...), the views of the OSS community on the OSS;
- Well documented : search on Internet (Wiki, Forum, mailing list ...);
- The way the product is realized or the use of recognize standard to realize the product the team working on the project: the team working on the project (there are known people in OSS community who have good reputation), the views of the OSS community on the OSS, users opinions on these OSS (Forum, ...).

### **Q10: If not evaluated: Why did you not evaluate the criterion?**


*Answer:*

*In some case, it's costly to evaluate some criteria.*

### **Q11: How important is this criterion to you?**

*Answer:*

- The maturity : 8
- Support (documentation, ...): 8

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 68 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

- The way the product is realized or the use of recognize standard to realize the product: 8
- Robustness: 8
- Evolvability: 8

## INTERVIEW 4

### Organizational Information

Date of Interview	3/11/2006
Interviewer	CETIC
Start Time	
Context / Domain	OS e-business integrator

### Interview Questions

#### Q1: For which types of purposes/systems do you (would you) use F/OSS products?

*The company specialized in OPEN SOURCE solutions, and we also use (quasi exclusively ;) F/OSS products 'in house'.*


- a) We use F/OSS products mainly to 'build' solutions for our customers (integrator) in the e-business domain.*
- b) We also use F/OSS libraries, components or servers for development of software or service for our customers.*
- c) And eventually F/OSS servers and Internet infrastructure for our deployment.*

#### Q2: Which F/OSS products do you use? (Have you considered any concrete products you would use?)

- a) for example: OS CMS products (Joomla, Drupal, Typo3, Jahia, ...), OS e-commerce products (OS Commerce), OS reporting (JasperReports),...*
- b) Ex: feed manipulation libraries for Podcast management software, Asterisk server for services (Web Telephony Integration),...*
- c) Linux servers with LAMP or OS J2EE environment + infrastructure management tools (Nagios, SubVersion server, OpenSSH client,...)*

#### Q3: How do you use F/OSS products?

- a) Integration + some modules dev. (sometimes core code modification but can be avoid for products with good architecture).*
- b) Integration with glue code + use of F/OSS libraries.*
- c) Use without modification, a set of servers that respond to our needs in term of infrastructure.*

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 69 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

#### **Q4: Why did you choose these F/OSS components?**

*Our company has choose to deliver OS solution because they have thefollowing advantages over commercial solutions:*

- access to source code (tailor made solution, adaptability)*
- perennality of the product (not dependent on a commercial company)*
- licence cost*
- ... to be completed...*

#### **Q5: To your knowledge, do you achieve business goals through F/OSS usage?**

*Since we belong to a public institution, we cannot strictly speak of business goals in our case. On the other hand, it is our goal to present our students with a good academic offer, and a well-organized, up-to-date web site is part of that. Arguably, our students are now better informed about our academic offer. We can also better support our lectures and seminars by making it easy for teachers and assistant to quickly publish information.*

*a) and b) YES, we succeed in a long series of projects based on OS products or components (see our website, in the 'reference' or 'case studies' section). We deliver 'tailormade' solution to our customers, solutions that fit their organizational needs; and they appreciate it!*

#### **Q6: What are criteria that F/OSS components need to fulfill to be appropriate?**

*Access to source code is mandatory due to our business (OSintegration).*

*a) criteria: -programming languages: PHP or JAVA -reliability -good references -documentation available (user, developer + comments in source code) -architecture of the software (modularity, evolutivity) -large community with recent activity -list of modules available and quality of these modules -quality of coding -support (for some projects)*

*b) idem a)*

*c)-References and notoriety-very stable products*

#### **Q7: How did you (would you) evaluate these criteria?**

*-reliability: discussion with other developers, Bug list (and resolution), forum discussions, past experience, reputation of the company who support the project, internal evaluation (tests...),*


*-good references: list of references + link on the official project website or in the community website.*

*-documentation available: browsing of the documentation*

*-architecture of the software (modularity, evolutivity): analysis of the architecture of the software (declared and real!)*

*-large community with recent activity: via OS projects statistics on websites like sourceforge.org, freshmeat.net, RFC lists, ...*

*-list of modules available and quality of these modules: development part of the website's project + forums,...*

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	Page : 70 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

*-quality of coding: direct inspection of source code*

*-support: type of contract support (respond time, pricing, part covered,...)*

## **Q8: Did you (would you) consider any criteria related to Robustness and Evolvability?**

YES.

## **Q9: How did you evaluate this criterion?**

*Usability: We looked at mailing lists to determine user satisfaction*

*Support: Look at activity in mailing lists*

*Robustness : see 'reliability', 'architecture', 'good reference' and 'quality of coding' in Q7*

*Evolvability: see 'architecture', 'list of modules' (numbers of recent modules and roadmaps) and 'community' in Q7*

## **Q10: (see Q7) If not evaluated: Why did you not evaluate the criterion?**

*Reliability: We did not know how to test the product for reliability*

## **Q11: QHow important is this criterion to you?**

*Rate on the following scale of 0 to 10, where 0 means that the criterion is irrelevant, and 10 means that a high quality level with respect to this criterion is critical for selecting the F/OSS component.*

*Usability: 8 (We have to give the product to untrained users, usability is fairly critical.)*

*Support: 10 (If we have questions, they need to be answered really quickly, because this usually means that our sales process could be stopped until the question is resolved)*

*Reliability: 7 (...)*


*It also depends partly on our clients priority! (as integrator). For example:*

*-for PE : support and robustness very important; large community not very important and J2EE based CMS => "Jahia CMS" -for RadioAmateurs : support and usability not very important; reliability, use of PHP and large Belgian community important => "Drupal CMS" -for ABCSoft : ecommerce product (with online payment => critical) with a lot of customization => references, reliability, list of modules available that respond to their needs and access to source code = important!! Quality of coding, documentation, architecture = not so important • "OS Commerce"*

## **INTERVIEW 5**

### **Organizational Information**

Date of Interview	November, the 16th 2006
Interviewer	FUNDP

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 71 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

Start Time	2.10 PM
Context / Domain	A public entity responsible for e-government in Walloon region. It promotes the use of OSS in public sector

## Interview Questions

### Q1: For which types of purposes/systems do you (would you) use F/OSS products?

*Answer:*

*We use OSS software in many cases:*

- Servers in operating system like Linux (mail, directory LDAP, sharing repository (Samba));
- Applications servers like APACHE, TOMCAT, MySQL, PHP, Linux, Python;
- Office applications like Open Office 2.0, Mozilla Firefox and Thunderbird;
- Tools Integration framework to build web applications like MediaWiki;
- Web Content Management Systems like SPIP;
- Management of projects like phprojekt;

### Q2: Which F/OSS products do you use? (Have you considered any concrete products you would use?)

*Answer:*

*We use OSS products like Linux, APACHE, TOMCAT, MySQL, Mozilla Firefox, Mozilla Thunderbird, MediaWiki, SPIP, phprojekt, Python, OpenOffice.*

### Q3: How do you use F/OSS products?

*Answer:*


*We use the OSS software like a commercial software. We use the product itself and sometimes, we modify some source code to adapt it with our needs. It was the case with phprojekt and SPIP. We also use libraries like python's libraries.*

### Q4: Why did you choose these F/OSS components?

*Answer:*

*We choose to use F/OSS components for several reasons:*

- To reduce cost;
- To promote the philosophy of OSS community
- To show and promote the use of OSS software in public enterprise
- There are many OSS software of good quality. It is the case of Linux Server for example (In one year, we stop our Linux Server for 2 min), APACHE
- To reuse source code of OSS

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 72 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

- For OSS stability
- Linux Server is the most robust server

### **Q5: To your knowledge, do you achieve business goals through F/OSS usage?**

*Answer:*

*Yes, we are very satisfied with OSS. They are very easy to use and free. The only problem we have is to exchange OpenOffice's format (.odf) into Word (.doc). Apart from that we do achieve our business goals with OSS.*

*We are working now on Belgif (the Belgian interoperability project) and we will promote the use of ODF (Open Document Format) in the Walloon Region.*

### **Q6: What are criteria that F/OSS components need to fulfill to be appropriate?**

*Answer:*

*To be appropriate, OSS components need to be:*

- available (source code, documentation, ..)
- Known as belonging to a group with an history (Version available, how do these version evolve ...). We refer to Sourceforge and other website for the history of some OSS.
- Popular (have an active community that contribute to the life of the software)
- Simply like MediaWiki
- In case we want to modify code, the OSS components need to be legible and readable, well documented, well structured and to have a strong and coherent architecture.

### **Q7: How did you (would you) evaluate these criteria?**

*Answer:*

*We evaluate them in specific projects or through some information from Internet. It could be:*


- The number of users of the OSS product in the community : are they active users ?
- The support (documentation, architecture, ...)
- Finding comparatives studies like CMS
- The views of the OSS community on the OSS product, his history
- The comparisons of functionalities : estimations of how well the OSS product fit our needs.

### **Q8: Did you (would you) consider any criteria related to Robustness and Evolvability?**

*Answer:*

*For Robustness, some criteria would be:*

- Maturity (life cycle, tests' results, ...);
- Powerfulness and stability of the product or the stack of software like LAMP (Linux-Apache-MySQL-PHP);

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 73 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

- Scalability
- The way the software could recover data after a problem

*For evolvability, some criteria would be:*

- The use of recognize standard to realize the product;
- The time taken to integrate a new technology;
- The structure of product's components (modularity);
- The history of the product;
- Documentation (Wiki, Forum, ...) ;

### Q9: How did you evaluate this criterion?

*Answer:*

*In many cases, we get the reputation of the product from Internet (the community working on, the story, the number of versions, his architecture, the views of OSS community, documentation, ...).*

*We can also ask a private enterprise to carry on comparative studies about functionalities of different OSS products and we choose the one that fits better our need.*

### Q10: If not evaluated: Why did you not evaluate the criterion?

*Answer:*

*Can not apply in our case.*

### Q11: How important is this criterion to you?


*Answer:*

- Robustness : 9
- Maturity (life cycle, tests' results, ...): 7
- Powerfulness and stability of the product or the software like LAMP (Linux-Apache-MySQL-PHP) : 9
- Scalability : 8
- The way the software could recover data after a problem : 6
- Evolvability : 8
- The time needed to integrate a new technology : 7
- The use of recognized standard to realize the product: 10
- The structure of the product's components are (architecture of the system) : 9
- The history of the product : 6
- Documentation (Wiki, Forum, ...) : 9

## INTERVIEW 6

### Organizational Information

Date of Interview	30/10/2006
Interviewer	URJC
Start Time	17:00
Context / Domain	OSS Business model

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 74 of 88</p> <hr/> <p>Version: 1.3 Date: Mar 15, 07</p> <hr/> <p>Status : Proposal Confid : Restricted</p>
---	--	--

## Interview Questions

### Q1: For which types of purposes/systems do you (would you) use F/OSS products?

*The company uses open source software (OSS) for different purposes. Their basic goal is to develop software. The company understands OSS as a philosophy, so they think in OSS as a business model.*

*OSS as a philosophy, so they apply this philosophy. The company has a community and they work as another developer. The company has contributed in different OSS they think are interested in. Their business model is based in a strategical point of view. work only with open communities and open software projects. They participate and contribute in OSS and the OSS community obtains advantages and The company as well.*

*In this moment we can consider different OSS infrastructure The company uses. Desktop, server, public website, community public website innovation*

### Q2: Which F/OSS products do you use? (Have you considered any concrete products you would use?)

*The main requirement they need to use software is that it has to have an important community behind it.*

*We have to differ between applications.*

*Server: Ubuntu and Debian.*

*Desktop: Ubuntu with GNOME.*

*Public website: In this moment they have migrated to typo3. Some months ago they used EzPublish.*

*Internal development: Jhbuild, Tutos, Tinderbox.*

*Other: developer environment, etc.*

### Q3: How do you use F/OSS products?

*--> QualOSS Usage Scenarios*


### Q4: Why did you choose these F/OSS components?

*They don't have specific requirements choosing software applications. The Company only wants software to be OSS and they don't like double license.*

*Some examples are GNOME versus KDE, Qt versus Typo3, etc.*

*KDE has, in some components, double license, so The Company can not access to the core of KDE. It is the same problem with Qt and Typo3. Qt has double license, so The Company can not develop Qt core.*

*It is also convenient that all OSS components we have talked about have an important community behind them.*

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 75 of 88</p> <hr/> <p>Version: 1.3 Date: Mar 15, 07</p> <hr/> <p>Status : Proposal Confid : Restricted</p>
---	--	--

*The Company doesn't like OSS which community is not active. Sometimes for strategic purpose they have to develop some software from OSS projects without a big community behind them.*

*In some cases they have to choose another specific software due to requirements. Some clients need different software.*

**Q5: To your knowledge, do you achieve business goals through F/OSS usage?**

*Their business model is based in OSS, so they need OSS to achieve business goals.*

**Q6: What are criteria that F/OSS components need to fulfill to be appropriate?**

*That question is similar to question number four. As main criterion, The Company would like to work with a really open community where they can develop software. They do not like double license, that is an important aspect.*

**Q7: How did you (would you) evaluate these criteria?**

*When The Company has to choose between different, first at all, that application has to be OSS. It is important other people thoughts about some products. We can underline the same said before. We like a lot an active community behind the projects we develop.*

**Q8: Did you (would you) consider any criteria related to Robustness and Evolvability?**

*It is important. If a product is not easily evolve or it is not robust probably we will need to develop it or simply we will not choose it. It is necessary, but we prefer to work with software less robust with a big community behind it.*

**Q9: How did you evaluate this criterion?**

**Q10: If not evaluated: Why did you not evaluate the criterion?**


**Q11: How important is this criterion to you?**

*In general they think different criteria are important, but the most important criteria is to have a big and open community behind the project.*

**INTERVIEW 7**

**Organizational Information**

Date of Interview	16.11.06
Interviewer	IESE

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 76 of 88</p> <hr/> <p>Version: 1.3 Date: Mar 15, 07</p> <hr/> <p>Status : Proposal Confid : Restricted</p>
---	--	--

Start Time	11:00
Context / Domain	E-Business

## Interview Questions

### Q1: For which types of purposes/systems do you (would you) use F/OSS products?

*The company does not produce products, but provides (web-based) information services on stock exchange data.*

*--> Offering http-based services; basically using Linux, TomCat, and Spring-Applications*

*We use OS products to provide services*

*A second area of the company is concerned with products (basically, clients for the stock exchange data). OS does not play a role there, except for usage in the infrastructure. The reason is that they program in Delphi; OS Licence issues are not a problem*

### Q2: Which F/OSS products do you use? (Have you considered any concrete products you would use?)

*Server/Plattform: To support the build prozess (Cruisecontrol), Apache, Subversion, Maiven (can generate Ant-Scripts)*

*MySQL, Linux (other types of operating systems, too);*

*Server: VRRPD (summarizes/maps several physical computers to a virtual address); Messaging-System (JMS-Server; Active MQ)*

*sometimes use of libraries; spring ([www.springframework.org](http://www.springframework.org)) (alternative to EJB), Logging*

*(Java-libraries)*

*The Wiki system we use is commercial. Reason: It is better in terms of functionality (rights management, PDF export, messaging), Look&Feel is better, and it was not too expensive*


*(Professional) Service/Support was not a reason to choose the commercial system.*

### Q3: How do you use F/OSS products?

*see above*

### Q4: Why did you choose these F/OSS components?

*selection of the systems: hearsay, experienced people watch newsletters an commjuncations. Some information comes from e.g. Java-Magazines, but this is minor. Most important is to participate in and watch the OS community.*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 77 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

*Previously, we had a tendency to program ourselves without looking for available systems. Today, we rather use what is there, in particular, since this means that we typically do not have to maintain and evolve it ourselves.*

*Evolvability is of moderate importance – the system has to be good already*

## **Q5: To your knowledge, do you achieve business goals through F/OSS usage?**

*Time to market*

*Amount of services*

*Business figures (revenue)*

## **Q6: What are criteria that F/OSS components need to fulfill to be appropriate?**

*The method is that we try it out and look at it in detail, including code.*

*--> [4-5] Readability/Analyzability: This is important, as we have to be able to read the code to evaluate it. In addition, readability can give trust into the reliability of the system*

*for example, we found a performance problem in our messaging system once. We had to debug the OS code and understand it to be able to find and correct the defect. It turned out to be correct parametrization of an OS component.*

*\*Documentation\* is important for readability*

*We have one programmer who evaluates candidate systems in detail*

*[9] Functionality: The product has to be suitable for the purpose. If the product evolves and gets better, fine, but this is not important*

*[7] Maintainability: Important, but not a major concern; we rarely have to correct defects in OS products. Sometimes, we patch systems, remove bugs, or implement features, but this is rare. The idea is to use products "out of the box" as much as possible. For understandability, the structure, dependencies, etc. are important.*


*Reliability: We have an own testing environment for our system. We try the OS product, and observe whether it has a negative influence on our software.*

*[10] Stability is the most important criterium; our systems have to be stable or we are quickly out of business*

## **Q7: How did you (would you) evaluate these criteria?**

*See above*

## **Q8: Did you (would you) consider any criteria related to Robustness and Evolvability?**

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 78 of 88</p> <hr/> <p>Version: 1.3 Date: Mar 15, 07</p> <hr/> <p>Status : Proposal Confid : Restricted</p>
---	--	--

### Q9: How did you evaluate this criterion?

*See above*

### Q10: If not evaluated: Why did you not evaluate the criterion?

*N/a*

### Q11: How important is this criterion to you?

*See above*

## INTERVIEW 8

### Organizational Information

Date of Interview	27 October 2006
Interviewer	CETIC
Start Time- End Time:	10:10AM — 12:45PM
Context / Domain	City Administration Management


### Interview Questions

#### Q1: For which types of purposes/systems do you (would you) use F/OSS products?

*Both of these are based on OSS infrastructure.*

*Answer from interviewee:*

- *Migrating servers from Unix SCO to Linux.*
- *Clients are currently Microsoft Windows but they all connect to the servers through the Metaframe to execute server side code. It is plan that client migration to Linux will be done at the end of 2007.*
- *Informix 4GL is used for the client GUI. It is now open source although we used it before it was open sourced.*
- *Since we have started a migration towards F/OSS with the integration of Zope and Plone for our workflow application, it is now much easier to migrate to other F/OSS products. We plan on migrating everything to F/OSS components one step at a time. This migration is actually possible because our expertise increase has we started to integrate our F/OSS components.*
- *Currently, we use MS Office and Outlook for mail but we plan on migrating to OpenOffice and a F/OSS mail client.*
- *Currently the DB server is still proprietary (from Informix) but we are currently but slowly migrating to Postgres.*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 79 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

- *We will also try through an exchange of expertise to include Asterisk (a VoIP service) in our website (the City's website) so that citizen can connect to the Administration via a Soft-phone on their computer.*

## **Q2: Which F/OSS products do you use? (Have you considered any concrete products you would use?)**

*Zope and a number of Zope products (add-ons).*

*Answer*


- *Zope, Plone and some new modules that we have developed on the top of Plone for the Administrative Workflow of the City Council activities such as booking meeting rooms, holding the Minutes of City Council meeting. Our code is also under a GPL and we are involved in CommunesPlone so others can start using and incrementing our code*
- *Apache is our webserver (we are not using the webserver from Zope but instead bypass it to go to Apache)*
- *Linux on servers (some servers are still Unix SCO but in three months they'll all be Linux)*
- *The compiler for Informix-4GL is now open source*
- *Postgres for DB server*
- *Asterisk will be integrated*

## **Q3: How do you use F/OSS products?**

*We use Zope as our main web server, extended with a number of OSS extension packages. We also wrote a number of scripts that do particular tasks, like creating a news bar at the main page, and tabulating the contact information for our group's members. Additionally, we created a Zope product that we use to support the workflow in some of our courses.*

*Answer*

- *Apache as our webserver*
- *Zope/Plone for the workflow segment of the application*
- *We have committers (write) rights for some core Plone modules*
- *We develop modules on the top of Plone for Administrative Workflow and for the web interface. We have also open our code.*
- *Zope, Plone and some new modules that we have developed on the top of Plone for the Administrative Workflow of the City Council activities such as booking meeting rooms, holding the Minutes of City Council meeting. Our code is also under a GPL and we are involved in CommunesPlone so others can start using and incrementing our code.*
- *Linux on servers*
- *In the next year, Asterix, OpenOffice and Linux on desktop*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 80 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

#### Q4: Why did you choose these F/OSS components?

*We needed a flexible system that would not require us to centralize the administration of our web server. This is possible with Zope. For example, at this time, when a group member is responsible for a lecture, it is possible for him to directly create and update the corresponding web pages without the risk of disrupting the rest of the server.*

*It was also convenient that Zope was freely available and had a large number of available extensions. The fact that I had previous experience with Zope also played a role in the decision.*

Answer

- *Actually prior to migrating to deciding Zope/Plone, Joel thought about integrating MS SharePoint Server however after conducting a series of interoperability test during 1 and ½ year where he integrated different workflow technology with the existing 200KLOC of Informix-4GL, the interviewee realized that Zope/Plone was the best one not only in term of interoperability but also because Python was a very clear language that enforces good programming practices.*
- *Migration to Linux was an easy choice since we already used Unix SCO.*
- *Other migrations to other F/OSS components has become possible because we have learned from the Zope/Plone and Linux experience. And given that we started a move to F/OSS we want to continue in that direction.*

#### Q5: To your knowledge, do you achieve business goals through F/OSS usage?


Answer

- *Yes, it allowed us to reach the objectives. We open a bid in where we listed two objectives and they were both reached.*
- *(1) To integrate the Plone workflow engine on the City's intranet.*
- *(2) For the consultant from BubbleNet to transfer his expertise on Zope/Plone but also on development technique such as pair programming.*
- *This second objective guaranteed we would be able to take over future development effort without the forced need for external consultancy.*
- *However, it is important to note that before the open bid, the interviewee had already selected the platform Zope/Plone. He had developed the city's website with Plone and had done the interoperability tests. All in all, he was very satisfied with Plone/Zope. Given he knew Plone, he could interview the consultant (who answered the bid) and he was able to select the consultant with the required level of knowledge (i.e., a real expert).*

#### Q6: What are criteria that F/OSS components need to fulfill to be appropriate?

Answer

- *The F/OSS components must be able to inter-operate in our environment.*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 81 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

- *The community behind a F/OSS component must answer our question so we 'll gain expertise in the F/OSS component*
- *I also was attracted by the clarity of Python code*
- *For the CMS, it was important that the technology chosen included a workflow components and that it included an development environment so that we could do some Rapid Application Development. Archetypes in Plone was exactly what I wanted. We can automate code generation from our UML models. We use Poseidon to create the model and generated the code thanks to Archtypes.*

### Q7: How did you (would you) evaluate these criteria?

Answer

- *For the interoperability test, I actually built a cluster of Zope servers and I was able to make our old Informi-4GL code interact with the Zope server and that Zope server could also exchange info with a second Zope serve that distributed the Internet pages.*
- *Independent of a technology being F/OSS or proprietary, I have always interacted with vendor/community who provided me with their technology. For example, even before Informix open source 4GL, I went to see them so they could explain certain thing about their technology. I have continued doing this for the open source product. In the case of Zope/Plone, I could interact with the community and also with the privilege partners of ZEA Partners.*
- *I also looked for coherence in my evaluation of technologies to integrate in my environment. For example to add a module to Zope, we simply need to call a function add with the new module.*


### Q8: Did you (would you) consider any criteria related to Robustness and Evolvability?

Answer

- *Yes, these two criteria*
- *The fact was that although I was not a Python programmer at first, the language it was very clear that Python would lead to very clear programs hence these programs would be easier to modify later on. We are in the process of integrating our development to Zope 3 which is quite different from Zope 2. It requires a lot of re-write in order to take advantages of the new mechanism of service interface however thanks to Plone Achetypes we can easily regenerate code hence we don't anticipate the migration to be too hard.*
- *Thank to Archetypes, evolving the code is much simpler.*
- *Robustness was definitely a criterion. Prior to taking the final decision to go with Plone, I created the city website and let it run for close to a year. It almost had no problem as compare to the previous city website in ColdFusion that crashed much more often*
- *More over, my interoperability test showed that the Plone/Zope alternative was more robust once integrated with our old application in Informix-4GL*

### Q9: How did you evaluate this criterion?

*Usability: We looked at mailing lists to determine user satisfaction*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 82 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

*Support: Look at activity in mailing lists*

*Answer*

- *The evaluation was done hands on and also by interacting with F/OSS communities e.g. the Plone community or the Informix-4GL community*
- *As a public body, we had the time to really evaluate the quality criteria related to robustness and evolvability. In our case, we tested robustness by letting the city's website written in Plone run over a one-year period.*
- *We also interacted with the Plone community and verified they were eager to communicate their knowledge so that we could later build an in-house expertise and master the solution we develop.*

### **Q10: If not evaluated: Why did you not evaluate the criterion?**

*Reliability: We did not know how to test the product for reliability*

*Answer*

- *They were evaluated so question skipped*

### **Q11: How important is this criterion to you?**

*Usability: 8 (We have to give the product to untrained users, usability is fairly critical.)*


*Support: 10 (If we have questions, they need to be answered really quickly, because this usually means that our sales process could be stopped until the question is resolved)*

*Reliability: 7 (...)*

*For this question, I (Jean-Christophe) suggested the different quality criteria below to the interviewee and he scored them.*

*Answer*

- *Functionality: 10*
- *Usability: 9 (by Usability, Joel specified that he meant the ability to master a product at all its level so that his team could later shape their development to meet any quality concerns. For example, modular solution to increase evolvability,*
- *Support: 9*
- *Reliability: 9*
- *Portability: 8*
- *Robustness: 8*
- *Evolvability: 7*
- *Performance: 5*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 83 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

- *Testability: 8 (we believe it is a important criterion for a successful open source solution and since we release our development on the top of Plone under the GPL so other cities can use it it, it is important we can write unit tests for all our development)*

## INTERVIEW 9

### Organizational Information

Date of Interview	06/11/06
Interviewer	MERIT
Start Time	11:00
Context / Domain	Health care / hospital

### Interview Questions

#### Q1: For which types of purposes/systems do you use F/OSS products?

*Time: 01:00*

*We only use it for firewall, a stripped down version of Linux. Besides, on a picture achieving system for radiology pictures, we have a server that sits between SAN (Storage Area Network) and PACS (Picture Archiving and Communication System), and that is also Linux based.*

#### Q2: Which F/OSS products do you use?

*The stripped version of Linux is arranged by Nokia who delivers the hardware, and organizes it with Check Point, which is the company for the firewall software.*

*We did also consider to use OpenOffice on the desktop, but if you look at the lock in and compatibility with other products we have it is impossible.*

#### Q3: How do you use F/OSS products?


*Very limited on the server side, not on the desktop side.*

#### Q4: Why did you choose these F/OSS components?

*We did in principle not choose ourselves to use open source; - or, for the firewall we did, because if you lay windows under here you already have a leak in your firewall, - but for the server between the SAN and PACS it is a choice by the company delivering the solution to us, so we had no choice. They said they always do Linux, so we had to take it.*

#### Q5: To your knowledge, do you achieve business goals through OSS usage?

No, there are not really OSS products for health care. Only OpenOffice, that I could think of considering for health care. If there were *real* OSS alternatives comparable to the available proprietary solutions in the market, it would be an alternative to be seriously considered for us.

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 84 of 88</p> <hr/> <p>Version: 1.3 Date: Mar 15, 07</p> <hr/> <p>Status : Proposal Confid : Restricted</p>
---	--	--

## **Q6: What are criteria that F/OSS components need to fulfill to be appropriate?**

*Good service, fulfill your requirements, continuity must be assured, and supported by supplier.*

*Highly depends on the product, OpenOffice should be integrateable with the whole hospital information infrastructure, and nobody supplies or supports this, while it is supported for Word. It is very hard to get it compatible with all the special software used in the organization.*

## **Q7: How did you evaluate these criteria?**

*Look at the supplier of the software, if they seem reliable. We want big firms with reputation. And you compare yourself to other hospitals, nobody wants to be the first implementing a new software package. The risk is too high if it goes wrong.*

## **Q8: Did you consider any criteria related to Robustness and Evolvability?**

*I would not know what that would be, specifically.*

## **Q9: How did you evaluate this criterion?**

## **Q10: If not evaluated: Why did you not evaluate the criterion?**

## **Q11: How important is this criterion to you?**

*Time:05:48*

*--> We followed ISO/IEC 9126/1 list as URJC did, to grade this and give reason to get some more real answers.*

*Maturity: 9 – very important it is stable. You need to assure the continuity. But it depends what you need it for. If it is critical for your processes it is very high, but obviously for less critical it is less important. We have own test-servers to test new software, and also for redundancy. For every application they in principle need a test server.*

*Fault tolerance: 9 (-- some unsureness if a high mark means accept a lot of fault or not --) for company critical processes, of real health care. Further, it obviously depends on how critical a processes is. For the health processes it is very important to have no error. For example, the quality of health care can be influenced by the software. If you have to send people home because you can not work.*


*Maintainability: 8- We would never want to do this ourselves, but it is very important.*

*Analyzability: Dont understand*

*Changeability: 7 - Can be standard.*

*Stability: 8 – critical, must be stabile, cannot fall out.*

*Testability: 7 – you test it once and you assume it is fine. Otherwise you will discover it with time.*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 85 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

*Adaptability: For users to learn to work with it? -- for the software itself, parameters? 8 - Upgrades/patches are very important.*

*Installability: 7 – you do not want to spend too much time on it and it should be possible to interpret it.*

*Coexistence: ? - 9 – it is a requirement, so possibly a 10. We run many applications.*

*Replacability: 6 – if it is a good product you dont change it so easy.*

*Safety: 9 – privacy and such. Safety, that everything you fill in is correctly recorded and nothing goes wrong that you do not discover. For example, if you record the blood group of someone, that it is nothing wrong in a routine so the incorrect numbers is given.*

## INTERVIEW 10

### Organizational Information

Date of Interview	26/10/2006
Interviewer	URJC
Start Time	11:00
Context / Domain	Services company

### Interview Questions

#### **Q1: For which types of purposes/systems do you (would you) use F/OSS products?**

*This is a company whose main goals are to get benefits from advertisements. Their business strategy consists of offering services. They are orientated to offer some services, on the other hand clients sometimes have to receive some publicity from the company and partners.*

*the company works with OSS for email platform. They have a big infrastructure, however they don't use OSS for all their platform.*

#### **Q2: Which F/OSS products do you use? (Have you considered any concrete products you would use?)**

*We have to differ between applications.*


*Server and Desktop: Debian and Red Hat.*

*Email server: Squirrel.*

*Data base: MySql.*

*Libraries: OSS Java libraries working over Microsoft platform (ISS) or Apache server.*

*Other services: OSS forums.*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 86 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

### Q3: How do you use F/OSS products?

*When we need to cover a new functionality in our platform we analyze the solutions, including F/OSS. We download the product, test it and if we are happy with it (it is stable and covers our needs) we deploy it in production.*

### Q4: Why did you choose these F/OSS components?

*the company does not have specific requirements selecting OSS. They like software they have found as good software. Sometimes "good software" means software which has been used and it has had good results in another companies.*

*At the moment, the company is migrating email platform from others countries. the company works with all countries from South America. Thus, they need a powerful email server due to potential million users.*

### Q5: To your knowledge, do you achieve business goals through F/OSS usage?

*the company earns money from advertisements in its services, like the web portal, forums or the foot of the email messages. . the company has agreements with big companies to sell their products. Apple is an example of it. In this moment, if you are a student you can buy laptops cheaper. So, the company sends advertisements to its clients and some clients accept offers from the company. This company earns money with OSS in that way.*

### Q6: What are criteria that F/OSS components need to fulfill to be appropriate?

*the company is a distributed company. It means the company works in some countries and technology department is distributed. In this moment the company tries to unify some services in Spain. the company Spain has determinated different criteria to get all information from countries. There will be a big email data base from users around the world and email services will be centralized. It requires a policy of migration and, of course, quality requirements.*


*There are different technologies they use for this purpose. In that case it is very important robustness and evolvability (and other characteristics) because they manage a big infrastructure and the company would not like to have surprises during the migration process.*

### Q7: How did you (would you) evaluate these criteria?

*There were criteria when they started to migrate the platform to Spain. First place, they made a criteria standardization. They looked for a very stable operating system with commercial support. They needed to build a cluster for email server. Red Hat was the company selected for that purpose.*

*They needed other software products as data bases. In this way they were searching and found MySQL as a solution. They needed velocity and stability in that case, and MySQL was selected.*

*the company platforms in the world had to use the same file system to start the migration process and the same version of MySQL.*

	Measurement Requirements Specifications  Deliverable ID: D1.2	Page : 87 of 88
		Version: 1.3 Date: Mar 15, 07
		Status : Proposal Confid : Restricted

### **Q8: Did you (would you) consider any criteria related to Robustness and Evolvability?**

*the company does not have critical systems. They offer services and they do not need extra functionality related to strong security, quick processes, etc. the company only has to be worried about personal data protection and other Spanish laws.*

*It is important robustness and evolvability. They think is more important evolvability. There are other types of requirements, such as managerial, political and, of course quality requirements.*

*Respect to evolvability, the company likes software easily updating. If a software changes its own design, such as data base manager or similar, in following versions. It is not a good point for this software and probably it will be rejected in future.*

*Robustness is not too much important. We have a criterion related to robustness. We prefer stability and evolvability before robustness. Probably next versions will be better and developers will take into account it.*

### **Q9: How did you evaluate this criterion?**

### **Q10: If not evaluated: Why did you not evaluate the criterion?**

### **Q11: How important is this criterion to you?**

*We were talking to the company about different criteria. They made a list of different possibilities and its marks.*

*We took as reference ISO/IEC 9126/1. This standard names some characteristics the company thought were interesting. Marks are over 10 points.*

*Maturity: 9 – the company thinks maturity is very important. They do not use tools which are not mature and they do not develop them. If they need something probably they will look for another tool on net.*

*Fault tolerance: 6 – That characteristic is not important. the company does not have critic services. Thus they do not need a strong fault tolerance policy.*


*Maintaniability: 9 – They look for tool whose maintaniability was very easy and comfortable.*

*Analizability: 9 the company does not use software which is packed. An example is an .exe program in Microsoft platform.*

*Changeability: 9 – the company business model is based in web services. They look for tools which could be change in a easily way. Sometimes they need to add a feature or to add their logo to some tools. It is very important to have multilanguage tools. the company works with Portuguese and Spanish speakers.*

*Stability: 10 – That is the most important characteristic they named. They do not have critical environment, but they look for having less and less problems with installed programs.*

*Testability: 8 – This is an important characteristic. If the company uses a software it means that software has been tested. If they can not test a software they reject it.*

	<p>Measurement Requirements Specifications</p> <p>Deliverable ID: D1.2</p>	<p>Page : 88 of 88</p>
		<p>Version: 1.3 Date: Mar 15, 07</p>
		<p>Status : Proposal Confid : Restricted</p>

*Adaptability: 6 – It is not important for them.*

*Installability: 8 – They do not dedicate a lot of time to install software. If it does not work they reject it.*

*Coexistence: 9 the company does not want heavy programs. They prefer light software and it can coexist with other programs in the same computer.*

*Replaceability: 7 – When a software is installed it will not be replaced for a long time. Thus, this characteristic is not very necessary.*