Sponsored through Framework Programme Sixth (Call 5) by

European Commission

Information Society Technologies

Document Information

Date : Jan 21, 10
Version: **1.1**
Pages : 105

Owning Partner:
CETIC

Author(s):
Jean-Christophe Deprez, Kirsten Haaland, Flora Kamseu

Reviewer(s):
URJC (review 1), FUNDP (review 2)

To:
European Commision

Purpose of distribution:

Submission of deliverable D4.1 to the European Commission

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

Printed
on 08/06/08 at 03:17:51 PM

Status:

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

Confidentiality:

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

Deliverable ID: **D4.1**

(new, more appropriate title)
Title:

Official Title: Metrics System and Advanced Quality Models

New Title: QualOSS Methodology & QUALOSS assessment methods

Deliverable of Task 4.1

Deliverable: D4.1

Title: Metrics System and Advanced Quality Models

Executive Summary:

When acquiring a software product in the proprietary world, people are concerned about the product but as importantly, they also want to know about the enterprise that produces the software product, in particular, to verify its reputability and determine risks of conducting business with that enterprise. Such aspects are also present in the F/OSS world. However, the traditional notion of an enterprise producing a software component does not exist. Instead, we define the equivalent notion: an *FlOSS endeavor*. An F/OSS endeavor not only include the set of work products produced but also a set of community members, a set of software processes and a set of tools and dependencies on other F/OSS components.

Furthermore, assessing the reputability and risk of doing business with an enterprise can partly be expresses in the FlOSS context as assessing the robustness and evolvability of an FlOSS endeavor respectively, the capability of an F/OSS endeavor to solve current problems and to last in the future.

As a consequence, Section 4 describes the ***QualOSS methodology***, a flexible and rigorous approach for assessing F/OSS endeavors. To remain very flexible, the QualOSS methodology stays at a high-level. Its goal is to set generic requirements that QualOSS assessment methods must implement.

In turn, Section 5 develops the standard QualOSS assessment method and describes how that method respects all the requirements prescribed by the QualOSS methodology. The standard QualOSS assessment method enable assessing F/OSS endeavor for an acquisition context, called *full FlOSS collaboration*, where an enterprise plans to integrate a F/OSS component in a software product and also wants to keep the contribution channel open in both directions, that is, it wants to be able to contribute the F/OSS endeavor and also retrieve contributions from others if desired.

In the full F/OSS collaboration context, an important decision factor on whether or not a F/OSS component should be integrated in a software product relates to the robustness and evolvability of an F/OSS endeavor. Moreover, various types of employees in an enterprise have an interest in the assessment results, for instance, product managers, project managers, architects, analysts, developers, testers and even technical writers. In turn, the standard QualOSS assessment method answers questions of interest to each of these roles and evaluates assessment goals by aggregating answers to these questions. The assessed goals identify and evaluate the risks related to the robustness and evolvability of an F/OSS endeavor. Annex A presents the quality model and all the sub-goals assessed in the standard QualOSS assessment method.

The standard QualOSS assessment method is expected to take an effort of a few person-days to obtain the assessment results for an F/OSS endeavor. In some situation, it is however possible to exploit the existing results, if such results are shared. This is done by following a lightweight QualOSS assessment method. Section 5.2 describes how to create a lightweight QualOSS assessment method.

Conversely, in some high-stake F/OSS acquisition context, it is warranted to create new QualOSS assessment methods that go beyond the standard QualOSS assessment method. Section 5.3 presents a guideline on how such heavyweight QualOSS assessment method may be created.

CHANGE LOG

Ver.	Date	Author	Description
0.1	04/24/08	Jean-Christophe DEPREZ	Initial skeleton for deliverable D4.1
0.2	09/11/08	Jean-Christophe DEPREZ	Sections 1, 2, 3, 4 reviewed by Jacques Flamand
0.3	09/12/08	Jean-Christophe DEPREZ, Flora Kamseu and Kirsten Haaland	Sections 1, 2, 3, 4 and 6 ready for Review by URJC (still missing Section 5 and references and correct citations). Section 6 was co-written by Flora, Kirsten and Jean-Christophe
0.4	09/20/08	Israel Herraiz	First review by URJC: Sections 1, 2, and 3 reviewed
0.5	09/20/08	Jean-Christophe DEPREZ	Integration correction from review 1 by URJC. Unfinished Write Section 5
0.6	09/24/08	Jean-Christophe DEPREZ	Slight change in the definition of the QualOSS methodology (in part to include the concept of QualOSS assessment method) hence re-write of large portion of Sections 1, 2, 3, 4
0.7	10/03/08	Jean-Christophe DEPREZ	Added Requirements 3, 4 and 5 for the QualOSS methodology + finished Section 5.1 + Annexes A (partial), B (plan but not included yet), C, and D
0.8	10/06/08	Jean-Christophe DEPREZ	Wrote Section 5.2, 5.3, conclusion, exe summary and added references
0.9	10/10/08	Jean-Christophe DEPREZ	Added to Annex A + review intro of Annex A
0.10	14/10/08	Jean-Christophe DEPREZ	Integrated all of Annex A
0.11	15/10/08	Israel Herraiz	Review of sections 4, 5, 6 and annexes
0.12	15/10/08	Jean-Christophe DEPREZ	Review and accept correction from Israel
0.12.1	16/10/08	Flora Kamseu	Second review of sections 1 and 2
13	16/10/08	Jean-Christophe DEPREZ	Review and accept Flora's comment
13.1	16/10/08	Flora Kamseu	Second review of sections 3 and 4.1
1.0	16/10/08	Jean-Christophe DEPREZ	Sanity Check

TABLE OF CONTENTS

1 Introduction

1.1 Motivation

The strategic objective of the QualOSS project is to enhance the competitive position of the European software industry by providing a methodology and tools for improving productivity and the quality of software products. To achieve this objective, QualOSS notes that many organizations have started to integrate Free *libre* Open Source Software (F*l*OSS) in their systems. Currently, they acquire F*l*OSS product components based on ad-hoc approaches. It is therefore the aim of QualOSS to facilitate the acquisition of the most adequate F*l*OSS based on a more rigorous assessment methodology. Furthermore, this methodology must be flexible to answers the many questions raised by F*l*OSS acquisition of various business situations.

By *most adequate FlOSS*, many understand the most adequate F*l*OSS product, that is, a product that satisfies their functional and non-functional needs. However, this approach neglects a large issue of the software acquisition process. When acquiring a software product in the proprietary world, people are concerned about the product but as importantly, they also want to know about the enterprise that produces the software product, in particular, to verify its reputability. This factor is also present in the F*l*OSS world. However, the traditional notion of an enterprise producing a software product does not exist in F*l*OSS instead, we define the equivalent notion: an ***FlOSS endeavor***. Furthermore, verifying the reputability of an enterprise becomes ***assessing the robustness and evolvability of an FlOSS endeavor***[1]. In consequence, QualOSS aims to provide a methodology to assess the evolvability and robustness of F*l*OSS endeavors.

Methodologies for evaluating F*l*OSS such as the Open Business Readiness Rating (OpenBRR) and the Qualification and Selection of Open Source (QSOS) have emerged in the last couple of years. They seem adequate to obtain light assessment and select F*l*OSS applications where the risk of a wrong selection have little impact on an enterprise. However when the stakes of a wrong selection are high such as when selecting F*l*OSS components to integrate in entreprise software products, OpenBRR and QSOS lack thoroughness and rigor to be considered adequate. Accordingly, providing rigorous, trustworthy assessment results when stakes are high is an important goal of the QualOSS project. One particularly interesting F*l*OSS acquisition situation is when an enterprise has decided to integrate a F*l*OSS component in one of its application and also to collaborate fully by contributing to the F*l*OSS component as well as retrieve contributions from others if desired. First, this situation often induce high stakes for an enterprise and second, this scenario addresses most of the questions raised by other less demanding F*l*OSS acquisition situations. QualOSS therefore pays a close attention to this F*l*OSS acquisition scenario.

1.2 Strategy and Approach for Work Package 4: Re-thinking the QualOSS Methodology

The Work Package 4 is the core of the QualOSS project. It builds the complete version of the QualOSS methodology and specifies how to apply it. WP4 is divided in 5 tasks.

The initial work of Task 4.1 is to review the comments and validation on the prototype methodology created during WP1 and then determine how to adapt the QualOSS methodology to respond adequately to these comments. The comments on the prototype and the lessons learned from WP1 are:

- The usage scenario identified for WP5 would only minimally benefit from the type of assessment proposed in the prototype approach. Indeed, the E.C. review comments as well as our own internal and external validation of the outcomes of WP1 determined that the current prototype was built on a unclear usage

[1] Definitions of F*l*OSS endeavor, robustness of a F*l*OSS endeavor and evolvability of a F*l*OSS endeavor are given in Section 2.

scenario and therefore, real life scenarios that would benefit from the QualOSS methodology needed to be identified.

- During certain interviews with enterprise members during Task 1.6, people expressed that they could not grasp assessment results, in particular, they did not know if assessment results actually answered their questions and worries. This was mainly due to the fact that the goals and questions in the prototype were broad or unclear. Indeed, the prototype only gave definitions of quality characteristics and it was hard to know whether the measures and indicators used to evaluate a characteristics corresponded to the enterprises' viewpoint. Furthermore, rather than only obtaining few theoretically valid results, enterprises would prefer a flexible methodology that answers most of their questions using their view of risks related to F/OSS acquisition.

- Our internal review of Task 1.4 and 1.5 also concluded that assessment were disorganized, for instance, datasets for an F/OSS project were not always scoped at the same level and the assessment results aggregated information incoherently.

- We established that weakness of existing methodologies such as QSOS and OpenBRR is their lack of traceability in assessments. Our effort in Tasks 1.4 and 1.5 also had that weakness. In order to become more credible, it is important that assessments provide extensive traceability as it is done in other scientific and engineering fields.

- Several projects already provide quick help on F/OSS selection. First, a project such as Ohloh hosted at http://www.ohloh.net allows one to verify a series of basic information (or measures). Other assessment methodology such as QSOS or OpenBRR also helped to selection F/OSS applications when stakes are low and thus when little resources would be devoted to make the F/OSS selection decision. Finally, other projects such as FLOSSMETRICS and SQO-OSS already work on automating F/OSS dataset analysis. As a consequence, QualOSS would better serve industry by providing a flexible yet rigorous assessment methodology that answers industry's questions. Such a solution would have more added value.

Given the points above, a global re-thinking of the prototype was done during Task 4.1. As a result, it proposes a flexible *F/OSS assessment methodology in the form of a generic F/OSS assessment process* that can be applied in various F/OSS acquisition contexts. This generic process specifies the tasks to perform during an F/OSS assessment activity. For increase flexibility, the QualOSS methodology does not impose how to conduct the task but only the objectives to achieve. In practice, these tasks may therefore be conducted differently depending on the assessment goals, which are dictated by the business goals of an F/OSS acquisition. Nonetheless, Task 4.1 also proposes a standard way to apply the QualOSS methodology. In particular, it presents a standard QualOSS assessment method that respects the assessment process prescribed by the QualOSS methodology. Furthermore, Task 4.1 also explains how the standard assessment method can be customized to answer more advanced questions of more specific and demanding F/OSS acquisition situations. Such situations will be explored during our case studies in WP5. It is also worth mentioning that WP3 will follow the standard QualOSS assessment method for assessing F/OSS endeavor during Tasks 3.2 and 3.3.

Task 4.2 specifies how to answer the questions identified in Task 4.1 using indicators. An indicator, as proposed in WP1, is a function of several metrics on which thresholds are specified. These thresholds answers questions by assigning risk-levels from low to high. Furthermore, Task 4.2 also continues to develop methods and techniques to apply during the various task of an assessment activity. In particular, it develops an interpretation guide to help users understand the measures and indicators proposed resulting from the assessment of an F/OSS endeavor.

Task 4.3 is more of an explanatory nature and attempt to apply various AI and machine learning techniques to discover new risk indicators.

Task 4.4 applies the validation strategy propose during Task 4.1 on the F/OSS endeavors selected and processed during WP3.

Task 4.5 is a short task that updates the QualOSS platform. In particular, it implements the new indicators discovered and validated during WP4 which were not yet implemented during WP2.

1.3 Objectives of Task 4.1

The Description of Work (technical annex of the contract) anticipated that Task 4.1 would only need to augment the result of WP1 with advanced metrics and that Task 4.2 would create indicators based on these advanced metrics. However, based on our internal validation of WP1 and on the outcome of the year-1 project review, several shortcomings in WP1 results were pointed out. In turn, Task 4.1's objective slightly shift to address these shortcomings and solutions are propose in this deliverable D4.1. Below, the update objectives of Task 4.1 are listed, and an explanation regarding what objective address what shortcomings is given as well as a short summary of how each shortcoming is solved.

Objective 1: Task 4.1 must describe usage scenarios where the QUALOSS methodology is applicable.

The need for clear usage scenarios comes as an outcome of several problems identified during the validation task of Work Package 1 (WP1). Indeed, WP1 perfectly fulfilled its prototyping role in that it helped to analyze the problem at hand, and showed important weaknesses in the initially proposed QualOSS methodology . First, the prototyped version of the methodology failed to identify the actual object of interest of an assessment. In particular, the prototype decoupled the assessment of F/OSS product from that of F/OSS community and it did not clearly explain what combining these two assessments would mean. The second shortcoming is that concrete specific real world, business cases where F/OSS assessments are useful were not described in WP1. These two important shortcomings resulted in a confusion as to what the QualOSS methodology assessed and what real world scenarios would benefit from using the QualOSS methodology.

Task 4.1 solves these weaknesses in several steps. First, Section 1.1 above already pointed out that companies needed help in their F/OSS acquisition process. It further explains that during software acquisition, business organizations are not only concerned about a software product but also about the enterprise that produces the software product, and in particular, how reputable that enterprise is. Transposing these concerns to the F/OSS world lead to the definitions for F/OSS endeavor and the robustness and evolvability of an F/OSS endeavor, which is given in Section 2. Furthermore this solves the first weakness identified by WP1 since the object of interest of an assessment is now clearly defined as an F/OSS endeavor.

Section 3 solves the second weakness by presenting general real-world FlOSS acquisition scenarios where the QualOSS methodology would help.

Objective 2: Task 4.1 must propose a flexible yet rigorous assessment methodology.

Every context where F/OSS components are integrated in a software product has its own specificities. In turn, the QualOSS methodology must remain flexible to apply to as many context as possible but it must also be capable of handling specificities of a particular context if one desires. As a consequence, to obtain the flexible desired, we decided to present the QualOSS methodology as a generic F/OSS assessment process composed of 5 tasks. All details on this process are presented in Section 4. It is then possible to create assessment methods that respect the generic process to any level of specialization of desired. In particular, it is possible to design fairly generic assessment methods that applies to many F/OSS acquisition situations or conversely very specialized assessment methods applicable to one or a few F/OSS acquisition situations. However, developing new assessment methods is not cost effective for most contexts. In turn, Section 5.1 presents an assessment method that respects the assessment process prescribed by the QualOSS

methodology. It is considered as the standard QualOSS assessment method. It is particularly suited to help in F/OSS acquisition scenario where an organization wants to integrate a F/OSS component in one of its software applications and plans to collaborate fully with the corresponding F/OSS endeavor by contributing to it but also by retrieving others' contributions. As argued later, the assessment method developed for this specific situation answers a broad range of questions also relevant to many other F/OSS acquisition scenarios. In turn, the standard QualOSS assessment method proposed is also quite flexible.

Beside flexibility, another important attribute of the assessment methodology is rigor. In this context, rigor means assessment coherence, reproducibility, completeness. Rigor is worth emphasizing because it was a large weakness of the work conducted in WP1. The validation Task 1.6 showed that the prototyped methodology resulting from WP1 lacked completeness and that the assessment effort conducted in Tasks 1.4 and 1.5 were sometimes incoherent and the lack of rigor would guarantee reproducibility.

Objective 1 above indicates that the object of interest of an assessment, namely, an F/OSS endeavor, will now be properly defined in Section 2. In turn, this will solve the problem of incoherence in assessment results, which were due to the difference of scope in the various datasets used when assessing product and community as two different objects.

To ensure reproducibility, assessments must diligently record a trace between raw and processed data as well as the exact methods used to process the data. In addition to reproducibility, this trace will enable an precise debate on the results of an assessment when such results are shared with others. For the QualOSS methodology, rigor therefore requires that both, the assessment process prescribed by the methodology and the assessment methods that respect this process enforce and encourage the recording of all information related to data and data processing when conducting assessments of F/OSS endeavors.

Concerning completeness, a user must be convinced that assessment results achieve a high degree of completeness when obtained the QualOSS methodology. As presented in Section 4.2, the assessment process prescribed by the QualOSS methodology is based on the activity system framework from activity theory. This framework, which has been tested and refined for over a century, proposes a comprehensive way of monitoring human activities, and indeed, an assessment is a human activity. In turn, constructing the assessment process by covering all parts of the activity system framework guarantees to reach a high degree of complete.

Objective 3: Task 4.1 must propose advanced assessment methods applicable to real-world F/OSS acquisition situations.

As mentioned in the previous objective, Section 5.1 presents the standard QualOSS assessment method. This method respects the assessment process prescribed by the QualOSS methodology and it applies to a real-life F/OSS acquisition scenario where an enterprise potentially wants to integrate a F/OSS component in a software products and furthermore, has the intend to contribute to the F/OSS component and to benefit from others contributions.

In addition, Section 5 also explains how the standard assessment method can be modified to answer to the need of different, more specific real-world acquisition situations. In particular, it will propose identify the part of the standard assessment method that may be customized and then propose guidelines to do so. Actual advanced assessment methods will be crafted in WP5 where the two real-world case studies described in deliverable D5.1 will be conducted. It is worth adding that the standard assessment methods described in this deliverable will be executed during WP3.

Objective 4. Task 4.1 must present the validation strategy that will be followed during other tasks of WP4, notably during Task 4.4.

Although not originally specified in the description of work, this objective needed to be added to D4.1 or else, the validation strategy would not be described in any deliverable prior to conducting the validation task. The validation strategy used in WP1 and described in D1.2 will be used as input for proposing the new validation strategy of WP4. However, some of the aspects covered in WP1 are not to be addressed in WP4 but rather by WP5; in particular, WP4 should only be concerned with validating the correctness or appropriateness of the results produced by the methodology but not whether these results satisfy the user who applied the methodology, this effort is left for WP5. Additionally, WP3 also performs validation; however, it is only concerned with data validation and validating it through interview with F/OSS community members. On the other hand, the validation in WP4 is concerned with the validation of the methods used for analyzing data (advanced metrics and appropriate use of metrics in indicators). WP4 also validates the method proposed to interpret results of a QualOSS assessment, which will be described in an QualOSS user manual.

On a final note on objective of Task4.1

The original objectives of Task 4.1 stated in the original description of work (DoW) have grown substantially. In particular, Objectives 1 and 2, which were identified as weaknesses by the validation task of WP1, were not planned in the DoW. Solving these weaknesses required a mind shift as compared to the anticipated work for Task 5.1. Consequently, Task 4.1 required deep thinking, thorough analysis hence significant effort.

Incidentally, one of the original objectives of Task 4.1 listed in the DoW mentioned that D4.1 would list the advanced metrics used by the standard assessment method. However, their listing has been postpone to deliverable D4.2 resulting Task 4.2. The reason for this change is the following. During WP1, we realized that the process of identifying metrics and then creating indicators (formulae of metrics with thresholds used to answer questions asked in the quality models) was intertwined into a single calibration process. In particular, in WP1 these two activities were dissociated and it resulted in listing too many metrics that could never be used by indicators. In turn, we decided to list metrics as part of the calibration process, which is an objective of Task 4.2 and therefore, the metrics will only be listed in D4.2.

1.4 Challenges For Workpackage 4

The main challenge of WP4 is to remove the weaknesses of the prototype proposed in WP1.

On the one hand, the QualOSS methodology proposed must be flexible and on the other, we must also show how it can be applied to business scenarios commonly found when integration F/OSS components in a product, infrastructure or service. We believe that this challenge was addressed during Task 4.1, which produced this deliverable.

For Task 4.2, the main challenge will be to find a good balance between automating an assessment and recognizing where manual actions are needed to augment significantly the value of assessment results. Furthermore, a challenge shared between Task 4.2 and Task 4.4 will be to identify and validate the thresholds proposed for indicators. We believe that the validation strategy presented at the end of this deliverable D4.1 will provide the adequate means to help Task 4.2 and 4.4 in this challenge.

1.5 Structure of the Deliverable

This deliverable presents the result from Task 4.1. After this introduction, Section 2 defines the terms used in this document. Section 3 first presents other effort related to QualOSS in order to justify why the QualOSS methodology is designed as it is. Second, Section 3.2 presents business scenarios as well as scenario where an assessment covers different scope of data. Section 4 describes the general QualOSS methodology. In particular, it presents the methodology as a F/OSS assessment process composed of 5 tasks and describes the objectives and expected results of each task. Section 5 then presents the standard QualOSS assessment method as well as ways to executed it in a middleweight, lightweight or heavyweight

fashion. Finally, Section 6 describes the validation strategy used in WP4 as well as WP3, this joint approach to validation was required to avoid duplication of effort.

2 TERMINOLOGY

This section presents the definitions of the terms used through this document. To start with, the introduction mentions that QualOSS is concerned about evaluating F/OSS endeavor. In turn, we give a concrete definition to F/OSS endeavor based on an earlier version proposed in (Deprez et al., 2007).

An ***FlOSS endeavor*** is the undertaking of community members following software processes to produce work products related to one or more F/OSS components. Furthermore, a F/OSS endeavor also includes the software process support tools (such as version control or bug tracking systems) and the software libraries on which the F/OSS components depends.

Formally, we define a F/OSS endeavor as a tuple of 4 sets, *CM*, *WP*, *SP*, and *TL* where

- *CM* is a set of community members,
- *WP* is a set of work products produced by community members, including the source code of the F/OSS components
- *SP* is a set of software processes followed by community members to produce work products, and
- *TL* is a set of tools such as libraries used to compile or run the F/OSS components or support tools to automate part of the software processes such as a version control or bug tracking system

We note that the definition of a ***FlOSS endeavor*** is purposely left fuzzy so the ***scope*** can be adjusted to the specific needs. This fuzzy definition makes the QualOSS methodology more flexible, that is, applicable in more business scenarios (such as those presented in Section 3.2). The scope of an F/OSS endeavor can be adjusted based on the business goal and context. In some cases, the context will justify specifying the scope of an F/OSS endeavor at the level of a whole F/OSS project, while in others the scope need to be defined at a finer level such a specific subset of releases or a particular version of a F/OSS subcomponent. Yet in some cases, the context could be larger and consider a family of several F/OSS projects as part of one F/OSS endeavor.

To specify the ***scope of the FlOSS endeavor*** to analyze, one must give a list of elements in each of the four sets, *CM*, *WP*, *SP*, and *TL*.

In practice, the scoping of these 4 sets is done partly automatically. For example, to scope WP, one will search particular subdirectories to identify all the source code files to analyze, then one will manually identify the documentation files that part of WP. Moreover, one may decide to include test scripts in WP. To scope *CM*, there are several approach. One may simply want to say all community members who produced a work product (or a part of a work product) listed in WP. Concerning CM, in some cases, one may be interested in only a few important contributors and therefore only they would be listed in CM. Similarly, one may consider all or just a few of the software processes in SP. For example, one may only be interested in assessing the maturity of only the release management and change management process areas. Regarding TL, one may decide not to care about the support tools but only analyze the compile time dependencies hence, only the libraries used by the F/OSS components would be listed in TL.

The *important aspect of scoping an FlOSS endeavor* is the repeatability of an analysis. In other words, the scope should be specified accurately enough so that an analysis could be repeating in a reliable way and the same assessment outcome would result. The other viewpoint is that the scope should be defined accurately enough in order to help argue the validity of assessment results. In turn, this will also help avoid confusion when debating on the results of an analysis.

The QualOSS project is concerned about assessing the *robustness and evolvability of an FlOSS endeavor*. Below we therefore give definitions for both concepts.

The ***robustness of an FlOSS endeavor*** is the degree to which an F/OSS endeavor is capable to keep functioning when mishaps occur —a mishap may be internal or external to the F/OSS endeavor in question. For example, a bug being reported or a dispute among community leaders are internal mishaps. A technological shift or the appearance of a new competing F/OSS endeavor are examples of external mishaps.

The ***evolvability of an FlOSS endeavor*** is the degree to which a F/OSS endeavor is capable to remain viable in the long future.

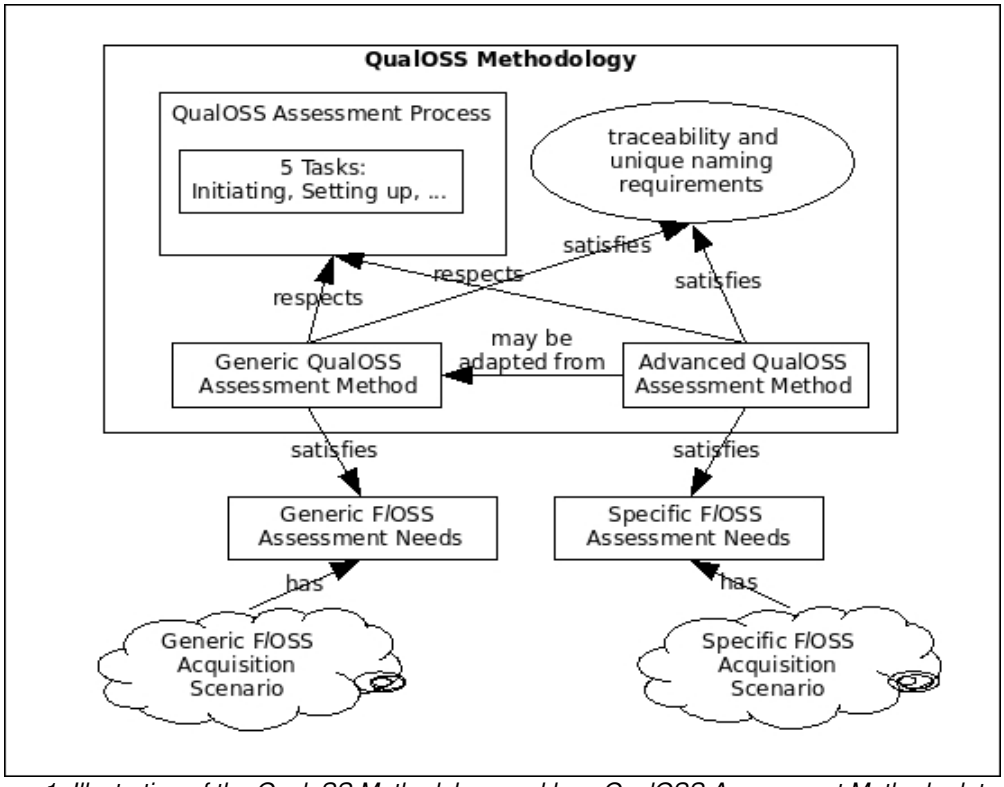

Figure 1: Illustration of the QualoSS Methodology and how QualOSS Assessment Method relates to it and to FlOSS acquisition scenarios. Arrow sources are the subjects, the arrows represent the verb relationships and the arrow targets are the objects.

The QualOSS methodology is developed to answer to the needs of F/OSS assessment, which are explicitly or implicitly connected to F/OSS acquisition scenarios or situations. Based on the Merriam-Webster online dictionary's definition of methodology[2], the ***QualOSS assessment methodology*** is defined as a body of ***QualOSS assessment methods*** used for conducting assessments of F/OSS endeavors. Furthermore, the QualOSS assessment methodology has *rules* (or requirements). They are specified in details in Section 4.1 An important rule concerns the assessment process that QualOSS assessment methods must respect. In particular, all QualOSS assessment methods must follow the ***QualOSS assessment process*** described in Section 4.2. In summary, the QualOSS assessment process breaks down an assessment in 5 tasks: (1) initiating an assessment, (2) setting up and planning an assessment, (3) collecting and analyzing data, (4)

[2] Methodology - 1 **:** a body of methods, rules, and postulates employed by a discipline **:** a particular procedure or set of procedures. (http://www.merriam-webster.com/dictionary/methodology)

interpreting the results, and (5) supervising an assessment. Each task defines its objectives and its outcomes.

A QualOSS assessment method is said to satisfy the QualOSS assessment process if assessment results (final and intermediate) obtained when following the QualOSS assessment method reach all task objectives.

Another important rule of the QualOSS methodology concerns the traceability of assessment results. In particular, it is important that a QualOSS assessment method enforces the traceability of assessment through built-in mechanisms and also by requiring that these mechanisms be followed (by software tools or by manual procedures) throughout an F/OSS assessment activity.

With regards to our objective of building a rigorous and flexible F/OSS assessment methodology, we may say that on the one hand, the requirements set by the QualOSS methodology, and in particular, the traceability requirement, help guarantee *rigor* in QualOSS assessment methods and how such methods will be followed to create their assessment results. On the other hand, the assessment process imposed by the QualOSS methodology is quite generic and therefore guarantees *flexiblity*. In particular, it is possible to build many QualOSS assessment methods. Each method would better fit a particular F/OSS acquisition context. As shown in Figure 1, a F/OSS acquisition context has particular F/OSS assessment needs. In turn, a QualOSS assessment method helps to obtain assessment results that satisfy these given needs. Furthermore, Figure 1 shows that a QualOSS assessment method must respect the QualOSS assessment process and also satisfy other requirements. Together, the QualOSS assessment process, the requirements and the QualOSS assessment methods represent the QualOSS methodology.

It is theoretically possible to build from scratch a new QualOSS assessment method for every specific F/OSS acquisition context. However, this would not be cost effective. Conversely, it is possible to construct a single generic QualOSS assessment method but it would only satisfy very few common F/OSS assessment needs hence would have little added value. This is somewhat the problem with current F/OSS assessment methodologies such as OpenBRR and QSOS, which are further presented in Section 3.

The goal of QualOSS is therefore to develop the standard QualOSS assessment method that is a good compromise between too generic and too specific. In turn, we decided to use the full F/OSS collaboration scenario presented in Section 3.2.1.2 as reference point to build the standard QualOSS assessment method presented in Section 5.1. Briefly stated, in the full F/OSS collaboration scenario, an enterprise plans to integrate a F/OSS component in a software product. Furthermore, this enterprise wants to be allowed to contribute the F/OSS endeavor that product the F/OSS components and also wants to be able retrieve contributions from others. This full F/OSS collaboration scenario is a F/OSS acquisition scenario that has many assessment needs. Most of these needs are also expressed in the other F/OSS acquisition scenarios presented in Section 3.2.1.2. In turn, the standard QualOSS assessment method of Section 5.1 can be followed to assess F/OSS endeavor based on the full F/OSS collaboration acquisition scenario.

Furthermore, it should also be possible to perform assessment for other F/OSS acquisition contexts with slight adaptations to the standard QualOSS assessment method. To this end, Section 5 briefly explains how the standard QualOSS assessment method can be adapted to work for other F/OSS acquisition contexts. Adaptations to the standard QualOSS assessment method will be explored during the cases study work package (WP5) where applications of the QualOSS methodology on real world F/OSS acquisition contexts will be studied.

In order to propose assessment results with high added value, the standard QualOSS assessment stays focused on assessing the robustness and evolvability of an F/OSS endeavor. Arguably, it is a significant portion of the F/OSS assessment needs in the full F/OSS collaboration scenario.

A efficient technique for assessing high-level concepts such as robustness and evolvability is to subdivided them into simpler concepts and repeat the division until the derived concepts become simple enough to be

evaluated. This division of concerns creates an tree hierarchy, if the leaf of the tree hierarchy are quality characteristics, then it is referred to as a ***quality model***. Figure 2 illustrates a fairly comprehensive quality model for defining *robustness and evolvability of an FlOSS endeavor*. Depending on one's desire, the leaf may be refined further or other quality characteristics added. Annex A present the quality model used in the standard QualOSS assessment method. Furthermore, Annex A also gives the definitions of these characteristics.

The goal of an assessment method is to assign scores to each of these leaves and eventually, aggregate these scores higher up the tree of the quality model. To obtain assessment results with high added value, that is, results that address the specific needs of an F/OSS acquisition scenario, it is important to tailor the assessment of each leaf characteristic to the particular needs of an F/OSS acquisition scenario. For instance, the emphasis on how to assess community regeneration in the context of the full F/OSS collaboration scenario may not be the same thing as in other F/OSS acquisition contexts. In the standard QualOSS assessment method, instead of merely using the generic definitions of characteristics, we complement them with comments that explains how each characteristic should be understood and assessed in the given full F/OSS collaboration context. Thus, the definition of a characteristic and the additional comments define the assessment goal. In this way, we may say that our approach follows the Goal-Question-Metric paradigm (GQM) (Basili, 1992).

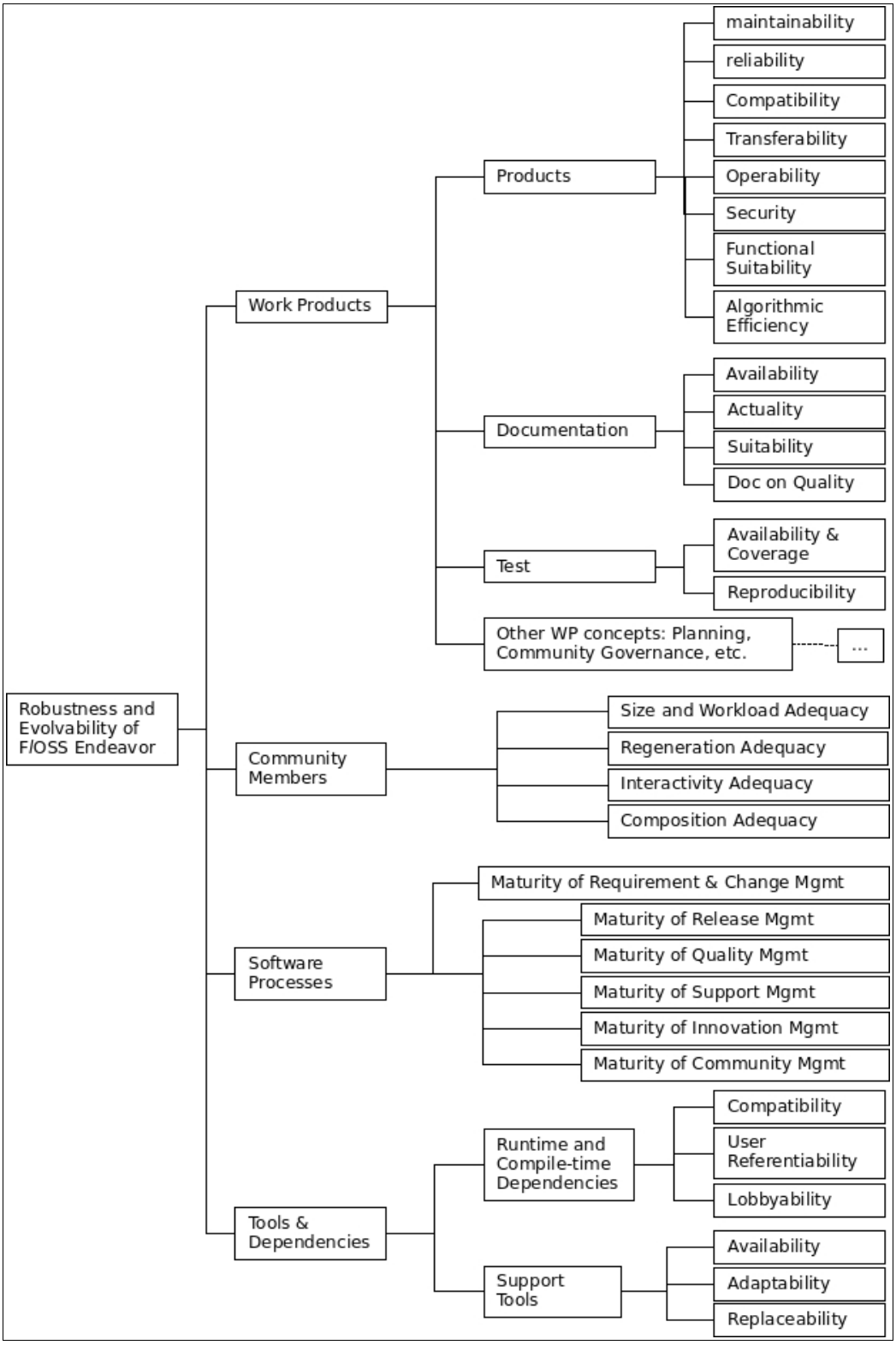

Figure 2: Sample Quality Model of characteristics that determine the robustness and evolvability of an FlOSS endeavor.

GQM first asks to derive the assessment goals from business goals. In our context, the business goals are defined by the full F/OSS collaboration scenario. By complementing the definition of a quality characteristic with comments specific to the full F/OSS collaboration scenario, we therefore express the assessment goals specifically targeted at the needs of the business goal. Annex A displays the definition of each characteristic in the quality model used in the standard QualOSS assessment method and also adds comments to explain how a characteristic is to be understood in the full F/OSS collaboration scenario.

The second step of GQM is to identify a series of questions for each assessment goal. The combined answers to these questions will help determine the degree to which an assessment goal is satisfied. Annex A also presents the list of questions related to each assessment goal.

The next step of GQM is to determine how to answer each question and how answers can be combined. The approach proposed is to answer questions using sound data analysis and sound measurements. Next, the results of the answers is combined using indicators. In the QualOSS project, indicators are being developed to quantify the perceived risks associated to assessment goals, for instance, a predictable behavior in a F/OSS endeavor will be seen as less risky than unpredictable actions, even if such actions may sometimes generate great outcomes.

In particular, we note that enterprises prefer to have risk-based scores that address their particular concerns rather than scores theoretically sound but not directly related to their questions of interest and often hard to interpret. For example, giving a score to product maintainability using the maintainability index formula from the scientific literature is perceived less useful that a score derived from answers to understandable questions such as “how many bugs were reported in the last stable release of a F/OSS component?”, or “what is the percentage of comments in the code of a FlOSS component?”

Indeed, most enterprises prefer to quantify how predictable a behavior is rather than how good it is because it is too often improvable to show that a particular behavior yields better results than another. Simply because in human intensive activities, there are probably no behaviors that is best for all situations. There is always a trade-off between rigor or methodical behavior, which takes time, and quick but less reliable behaviors. Thus, when assessing characteristics of an F/OSS endeavor, which is an human intensive activity, the best approach, at least from an enterprise's viewpoint, is to quantify whether an F/OSS endeavor is undertaken in a predictable way.

A ***QualOSS indicator*** is therefore an instrument that aggregates sound data analysis results to evaluate the risks associated to a quality characteristic of a F/OSS endeavor for a particular F/OSS acquisition scenario. Furthermore, in the standard QualOSS assessment method, QualOSS indicators are created to assessing the risk associated the robustness and evolvability of an F/OSS endeavor for the full FlOSS collaboration acquisition scenario described in Section 3.2.1.2. Task 4.2 is currently working on creating indicators to answer the questions raised in Annex A. The result of that effort will be fully presented in deliverable D4.2.

3 General Usage Scenarios for the QualOSS Methodology And Related Works

The main objective of this section is to present the scope of application of the QualOSS Methodology, that is, situations where applying the QualOSS methodology provide useful results. These usage scenarios are presented in Section 3.2. These usage scenarios remains fairly generic. It is only in Section 5 that specific scenarios are presented with the details on how to apply the QualOSS methodology to these scenarios.

Prior to presenting the generic scenarios, Section 3.1 presents works related to the QualOSS methodology. This in turn helps to understand how to design the QualOSS methodology to differentiate it from existing results as well as to identify how to leverage on existing results from these research effort.

3.1 Related Works

The effort presented in this section were for most already presented in deliverable D1.2, which presented a inventory of research results connected to QualOSS. So, unlike deliverable D1.2, this section does not present a full description of these other works but only summarizes how the QualOSS methodology can differentiate itself from these works or how it can leverage on these existing results .

The various efforts related to QualOSS may be categorized in two: (1) research specifically addressing F/OSS quality issues and (2) generic research on software product quality independent of whether the license is F/OSS or proprietary. The former category is developed below while research efforts in the latter category are already found in deliverable D1.2 hence only a short argument summarizes how the QualOSS methodology will make use of the works of the second category in the next paragraph.

Concerning software product quality, the QualOSS methodology plans to refer to and follow the ISO initiative, i.e., SQuaRE 25000 formerly ISO9126. The new standard 25000 series is currently revising the quality model of ISO9126 presented in D1.2. The document 25010, which belongs to the 25000 standard series, presents a new quality model for software product. This document is not finalized yet but it is well on its way and little modification to it are expected. Anyhow, Jean-Christophe Deprez from CETIC, involved in QualOSS, is the ISO Belgian representative in the working group drafting the 25000 standard series. In turn, he has access to the most current version of the revised quality model and will be notified of the small adaptation to the quality model.

In the category of research specifically addressing F/OSS quality issues, a distinction is made between (a) efforts that propose F/OSS assessment methods and (b) those that collect and process data with no intent to use the results for assessing F/OSS.

Table 1: List of research projects addressing FlOSS quality issues

Research related to F/OSS	
Assessment Methodology Projects	Data Collecting Projects
1. OpenBRR, 2. QSOS, 3. SQO-OSS	4. FLOSSMETRICS, 5. FLOSSMOLE, 6. OHLOH

The next two subsections 3.1.1 and 3.1.2 respectively explain how the QualOSS methodology is different from the projects 1, 2, and 3 and how the QualOSS methodology plan on leveraging from results from project 4, 5, and 6.

3.1.1 FLOSS Assessment Methodologies

The first two projects in the list, (1) OpenBRR (OpenBRR, 2005) and (2) QSOS (QSOS, 2006) propose a fairly similar approach for assessing F/OSS. A detailed comparison of the two methodologies may be found in (Deprez, 2008). One important difference between the two is that QSOS assessed version of a product and its surrounding context while OpenBRR principally aims at assessing whole projects. Both methodology are somewhat lightweight in that they propose simple manual scoring procedures that most IT people could follow.

These two assessment methodologies are adequate to help end-users decide what F/OSS applications to install and use. However, due to the lightness of their approach, these two methodologies are not thorough enough to help select an F/OSS component to integrate in a broader software product. Being more thorough to help in the selection of the best suited components to integrate in a larger software solution is the main aim of QualOSS. When questioning what F/OSS components to insert in an software product, the stakes are higher. A user of the assessment methodology wants answers to questions not addressed by OpenBRR and QSOS. Furthermore, the two existing methodologies do not analyze the quality of the software product in details. This concern is quite important when selecting a F/OSS component to integrate in a software product.

Besides these two initiatives, there is also (3) SQO-OSS, an European project, which runs in parallel to QualOSS. From the SQO-OSS description of work, the project is only focused on product quality and it is not concerned in assessing other aspects of an F/OSS endeavor. So unlike the QualOSS methodology, SQO-OSS does not evaluate the F/OSS community and its members, the software processes and the tools and libraries.

QualOSS plans to develop a methodology to assess a whole F/OSS endeavor. Comprehensively studying work products (including code, documentation and test), F/OSS community members, software processes and tools and libraries is required to obtain all the required information in order to make an informed decision when in the process of acquiring software. As argued previously, in a software acquisition process, the acquirer is not only concerned with the quality of the product but also the seriousness of the whole enterprise producing it. In consequence, as an acquirer questions the seriousness of the enterprise of the software product to acquire, in the case of FlOSS acquisition, he wants to know the seriousness of F/OSS endeavors. Two important aspects of serious F/OSS endeavor are robustness (ability to function when mishaps occur) and evolvability (ability to remain viable in the long future).

In conclusion, QualOSS recognizes that QSOS and OpenBRR provide adequate lightweight methodologies to help end-users select an F/OSS application. In both cases, the effort of evaluating a F/OSS project or a F/OSS project version takes at most a few hours. This is usually the time limit that small and medium enterprises are willing to invest. In such a short time frame and with limit expertise on configuring measurement tools and on interpreting results appropriately, it is unlike that SME staff would benefit from a new more thorough assessment methodology. Hence it is not QualOSS's intend to provide a methodology for these scenarios. Instead and as discussed further in Section 3.2, one of the usage scenario is to help assess a F/OSS endeavor in order to identify the most suited one so as to integrate its F/OSS component in a larger software application.

Regarding the selection of end-user application, it is worth noting that, in the event where the QualOSS methodology was already applied on a F/OSS endeavor, it may then be feasible that SME would benefit from these existing results. In that case, the only effort to invest by a SME would be to learn how to interpret the results and check that they cover all the aspects important to the SME's context.

Regarding the use of SQO-OSS results, QualOSS could have planned to use the tools developed by SQO-OSS for evaluating product quality. However, there is a high risk involve to plan on using results only available in the future. Furthermore, due to specificities in both descriptions of work, of QualOSS and SQO-

OSS, it is unclear that SQO-OSS tools would fulfill all QualOSS requirements for example, QualOSS must be able to treat software developed in Python and Ada and it is unclear if SQO-OSS tools would treat these two cases. In turn, the QualOSS consortium decided that it would implement it own methods for evaluating software product quality.

3.1.2 FLOSS DATA COLLECTION

Three projects are currently concerned with collecting F/OSS related data, namely, FLOSSMole (http://ossmole.sourceforge.net/), Ohloh (http://www.ohloh.net), and FLOSSMETRICS (http://www.flossmetrics.org). We briefly describe each project below.

The first comprehensive initiative in the area of F/OSS data collection is FLOSSMole. Its aim is to provide a database of information collected from well-know forges, and in particular, SourceForge. For SourceForge, a new snapshot of data is provided every 2 months. In addition, data from other forges are also collected, e.g., FreshMeat, RubyForge and ObjectWeb. FLOSSMole also accepts data donation from other project.

The main problem with using data from forges is that there are often incomplete. In many cases, F/OSS projects only use a forge for exposing their releases, that is, packaged distributions of their releases, for example, in the form of a zip file. For all other purposes such as issue/bug tracking or version control, F/OSS projects often decide to roll out and administer their own systems. Given that FLOSSMole does not collect data beyond those directly accessible on a forge, most of the data of F/OSS projects are missing.

In conclusion, due to the variable quality of F/OSS project data found on forges, QualOSS decided not to use SourceForge data. Furthermore, many interesting FlOSS projects are not present on the main forges, for example, those of the Apache Software Foundation or of the Eclipse Foundation.

Another project that collects and process F/OSS data is FLOSSMETRICS. Like FLOSSMole, FLOSSMETRICS share publicly the dumps of FlOSS data gathered and processed by FLOSSMETRICS. Unlike FlOSSMole, FLOSSMETRICS does not necessarily collect raw data from a forge but from other repositories that might be available. For example, FLOSSMETRICS collects version control data wherever available, that is, from the url of a F/OSS project site directly or of forge. Furthermore, FLOSSMETRICS process data in a deeper fashion than FLOSSMole, for example, by extracting data from version control repository of a F/OSS project. However, FLOSSMETRICS only collects data from the main line of development in version control repositories and currently does not process release branches.

As mentioned previously, one of the focus of the QualOSS methodology is to help select F/OSS components to integrate in larger software application. So in addition to obtaining data collected from the main line of development, it will also need data from release branches.

Ohloh is a third project that collects F/OSS data. However, unlike FLOSSMole and FLOSSMETRICS, Ohloh has created a website to facilitate the viewing F/OSS project data by human. Furthermore, it also gives an API to access its data in an automated and transparent way. The main drawback from using Ohloh data is that the original source of where the data was collected in not explicitly mentioned and in turn, data validity would be refutable and no argument could be given in defense.

In conclusion, QualOSS currently decided only to use FLOSSMETRICS data as-is. In addition, it will use tools produced by FLOSSMETRICS such as CVSAnaly (http://cvsanaly.tigris.org/) and MailingListStats (http://flossmetrics.org/sections/tools/MailingListStats). Furthermore, Ohloh has released Ohcount (http://labs.ohloh.net/ohcount) under the GPL license so QualOSS may decide to use it, in particular for it ability to identify the F/OSS license present in source code files. If Ohloh is willing to disclose additional information on where it extract its raw data then QualOSS may decide to rely on it. As Ohloh provides API to get access to its results, the overhead of accessing it would be very low.

3.2 Usage Scenarios For the QualOSS Methodology

The QualOSS methodology is defined to complement and leverage on the existing effort described in Section 3.1. As already stated, QSOS and OpenBRR in conjunction with Ohloh provide an adequate help to select F/OSS application when stakes are low, for example, to help an SME select a bug-tracking system to set up for internal use on their infrastructure. In such situation, an enterprise is willing to spend at most a few hours of an employee's time in order to select the appropriate F/OSS application. On the other hand, when stakes are high, companies are willing to spend more time and even contract out assessments of F/OSS endeavors. Stakes are higher when the situation touches to the core business of an enterprise, for example, when the product that will integrate the selected FlOSS component will be distributed to large amount of people, in particular, if that includes customers outside the enterprise. It is when stakes are high that the current methodologies are too shallow to be considered adequate. Hence, it is with these cases of high stakes in mind that the QualOSS methodology is developed and its application will be best suited.

During WP1 task 1.2, we interviewed several organizations that for most, already integrated the usage of F/OSS components in their infrastructure, their products or services. Based on their past experience of the interviewed organizations, we noted that they invested substantial amount of time selecting the best suited F/OSS components. Furthermore, during this time of investigation, they exchanged with the F/OSS communities in question to verify their reactivity and their willingness to collaborate.

In addition, interviewees who are planning to launch F/OSS endeavors recognized that they would benefit from advice on how to set up their F/OSS endeavor to increase its chances of success, i.e., advice on how to set up the F/OSS endeavor to maximize its robustness and evolvablility.

From the interviews conducted in WP1, we therefore identified two high-level scenarios where the QualOSS methodologies would best fit: (1) a comparison scenario where several F/OSS endeavors are compared for the purpose of selecting the most suited one and (2) an introspection scenario where an actor in a F/OSS endeavor wants to monitor the robustness and evolvability of his F/OSS endeavor. In the next two subsections, we describe these two scenarios further.

3.2.1 Comparison Scenarios

An F/OSS endeavor is a fuzzy term hence the comparison scenarios vary depending on how the scope of a F/OSS endeavor is defined. Below, we give four examples where the scopes are respectively defined at the level of (1) single F/OSS project, (2) several F/OSS projects, (3) subcomponents of F/OSS projects and (4) versions of a F/OSS project (or even versions of subcomponents of a F/OSS project).

Beside proposing scenarios based on the scope of FlOSS endeavor, it is also important to present business scenarios where F/OSS components are integrated by an enterprise and then explain why these scenarios require assessing the evolvability and robustness of F/OSS endeavors.

3.2.1.1. Comparison Scenarios at Various FLOSS Endeavor Scopes

Single FlOSS project

One wants to select among several F/OSS components to integrate in a larger software product. The final selection will not only depend on the F/OSS component but also on how robust and evolvable each F/OSS project is.

For example, an enterprise wants to build a new kind of softphone that works on PBX infrastructures. For this tasks, the enterprise wants to build its software product on the top of an existing F/OSS libraries. There already exists several open source projects that provide libraries for PBX, for instance, Asterisk, OpenPBX, CallWeaver, Yate, sipXecs, and FreeSwitch. The enterprise wants to identify which of these projects is the most robust and evolvable F/OSS endeavor for integration in the future softphone solution.

In the above scenario, the scope of the F/OSS endeavor is defined at the level of F/OSS project, in particular, each of the F/OSS projects listed above would be considered an F/OSS endeavor and the assessment exercise would therefore evaluate each of these F/OSS endeavors in relation to the softphone context. Clearly, the current softphone context is not fully specified. For example, it is neither specified whether the new softphone will be released under an open source or a proprietary license nor whether the enterprise consider contributing to the selected F/OSS endeavor. Depending on these variations, the assessment would also likely vary. More information on more precise business context will be given in Section 5. In the meantime, below, we give a few examples of how an assessment may vary depending on different business goal for using an F/OSS component.

If the enterprise merely wants to fork the code and modify it into a proprietary product without contributing changes back to the F/OSS endeavor then clearly, the assessment will mainly focus on code quality and will mostly ignore the aspects tied to the community, the processes and the tools of the F/OSS endeavors. On the other hand, if the enterprise wants to contribute to the F/OSS endeavor for example by contributing new code and new contributors then it will be important to assess the four aspect of an F/OSS endeavor work products, community members, software processes, and tools. In other cases, the enterprise may only want to integrate an F/OSS component without modifying it but leaving the opportunity to upgrade to new versions of that F/OSS component in the future. In this last scenario the assessment will not focus much on code quality but most of its emphasis will look into API or user documentation and also on community and its software processes.

The scenarios above consider an F/OSS endeavor at the F/OSS project level and are therefore fairly intuitive. However, as shown below, they are also other scenarios where F/OSS endeavors must be defined as a set of F/OSS projects or as a part of an F/OSS project.

Sets of FlOSS projects

One wants to select among several family of F/OSS components to integrate in a larger software product. The final selection will not only depend on the F/OSS components but also on how robust and evolvable each family of F/OSS projects is.

For example, an enterprise wants to build an advanced web application for Business Intelligence reporting. For this purpose, they are considering using Pentaho with Eclipse BIRT or develop a home-made advanced web interface on the top of JasperReport and iReport.

In this scenario, Pentaho and Eclipse BIRT would be considered as the first F/OSS endeavor while the second FlOSS endeavor would be composed of JasperReport and iReport. The assessment exercise consists of evaluating these two endeavors to identify the one most suited to the situation. It is worth noting that the assessment of a family of F/OSS projects is not identical to assessing each F/OSS project in isolation only. In particular, they are additional factors to consider. For example, when assessing the set JasperReport and iReport we would also check whether or not the F/OSS communities of the various projects have defined an established channel communication, whether or not the software processes such as release management are sync up between JasperReport and iReport, whether they are under the umbrella of the same F/OSS foundation or not, etc. All these matters would not be considered if JasperReport and iReport were assessed in isolation however, they influence the robustness and evolvability of the set JasperReport and iReport.

Subcomponents of FlOSS projects

One wants to select among subcomponents of several F/OSS projects to integrate the most suited subcomponent in a software product. The selection will not only depend on the F/OSS subcomponents themselves but also on how robust and evolvable each F/OSS endeavor producing each subcomponent is. Below, we present an example where F/OSS endeavors needs to be defined at the level of subcomponent.

An enterprise wants to build a new reporting GUI on the top of existing reporting engine. One of the existing engine considered is the Eclipse BIRT engine, which is only a subcomponent of the whole Eclipse BIRT project. There is difference in the assessment of a project sub-component versus a whole project. For example, there might be a different level of commitment on the various sub-component of the BIRT project. The activity on the BIRT engine may also be different from that of the BIRT GUI front end for instance. The documentation may also vary for the various subcomponents. For example, there may be no technical documentation on the BIRT engine while tons of tutorial on how to use the GUI may exist. Even the software processes may also be different, for example, the BIRT engine publishes a roadmap of its future development while the remaining subcomponent do not define (or at least do not share) such a roadmap. All these potential variations explain why it is important to distinguish between the assessment of a whole project vs. only a sub-component. However, this is not to say that the assessment of a sub-component should not also provide information regarding the project to which it belongs.

In the Eclipse BIRT Engine scenario, the F/OSS endeavor is scoped at the level of the BIRT Engine and not the whole BIRT so the main focus of the assessment would analyze the work products, community members, tools and software processes related to the BIRT engine and little to no emphasis will be put on other components of BIRT. An assessment of the BIRT engine will nonetheless have to address issues such as how independent is the BIRT engine endeavor from other parts of. This would for instance answering the following questions: Is the BIRT engine code well encapsulated and independent from other BIRT subcomponents?, is the community behind the BIRT engine strong enough by itself or is it heavily depend on the remaining of the BIRT community? etc.

Version of a FlOSS project

This last scenario is somewhat different from the others. Indeed, the first three types of scenarios are for selecting F/OSS code to use in a larger application. On the other hand, this scenario assumes that a F/OSS component has already been integrated in an application. The concern is now to know when to upgrade to new version of that F/OSS component.

This scenario is actually studied in one of our WP5 case studies and is described in D5.1. In particular, it concerns the migration to a new version of the GCC back-(part of the GCC compiler) in the GNAT Pro Ada Compiler. In this very case, Ada is not even concerned about the whole GCC endeavor but only about the GCC Core (or back-end), which takes a type-bound AST as input and generates optimized machine code for various hardware architectures. AdaCore's concern is to know when a new version of the GCC back-end is stable and reliable enough so it is worth migrating their GNAT Pro Ada Compiler to this new version.

In these kind of scenarios, a F/OSS endeavor is scoped at the level of a single version of software product or even, at the level of a single version of a sub-component of a software product, such as in the AdaCore's case where the concern is not a version of the whole GCC but only a version of the GCC back-end.

This scenario is different from the other three scenarios presented. In this last case, the various versions compared are from the same F/OSS project. Nonetheless, this kind of scenario also requires analysis beyond code. For example, observing the change management process may show that bug patches are provided quickly for the two most recent versions but not for the previous ones.

In the AdaCore scenario, the F/OSS endeavor is scoped at the level of a version of the GCC back-end. Hence, GCC-4.1 BackEnd is considered the FlOSS endeavor to analyze and its assessment results could be compared to GCC-4.2 BackEnd . While the main focus of an assessment may be on code quality and code variation between versions, the assessment will also need to answer questions related to (1) the community members who contributed to various versions of the GCC back-end considered, (2) the tools and libraries used in the GCC back ends, and (3) the software processes and how they were followed in each version.

3.2.1.2. Business Scenarios and their needs for FLOSS assessment

This subsection describes six common business scenarios where an enterprise wants to integrate F/OSS components in a product or an infrastructure. Furthermore, we also explain why the business scenarios benefit from an assessment of the robustness and evolvability of the corresponding F/OSS endeavors.

Although 6 scenarios are presented, it is our opinion that Scenario 1 is the best way to integrate a F/OSS component. All the other scenario miss certain opportunities offered by using F/OSS. However, we do understand that it is not always possible to enter in a full and open collaboration with F/OSS product due to the current shape of the market. Consequently, we provide other realistic business scenarios of F/OSS integration.

Scenario 1: Full FlOSS Collaboration

In this scenario, an enterprise completely understands and has integrated the F/OSS development process in its way of conducting business. It commonly wants to integrate F/OSS components in its products. Once the initial integration is done, the enterprise wants on one hand, to be able to integrate new changes proposed by others and on the other hand, it also wants to contribute its changes. Depending on the specifics of the integration scenario and the expertise of the enterprise, the enterprise contributions may vary from code, documentation, test effort, translation or even monetary support or computing and space support, for example, for conducting nightly builds and tests.

In this scenario, it is clear that when an enterprise selects an F/OSS component to integrate in its product, it wants to have a very good picture of the various aspect of the F/OSS endeavor producing that F/OSS components, and in particular, information regarding the robustness and evolvability of that F/OSS endeavor. For example, the enterprise wants to know what individuals and other companies are involved in the F/OSS endeavor as well as know about their level of commitment, it also wants to know how the community members interact (software processes). Furthermore, if the enterprise considers contributing code, it will desire to know about the quality of code and of other work products such as tests and documentation. Finally, it will need information on the dependencies between the considered F/OSS components and other F/OSS components. It is only with a full picture of an F/OSS endeavor that the enterprise will be in a position to make an informed decision on whether it is worth integrating a F/OSS component or not.

The standard QualOSS assessment method developed in Section 5 is specifically designed for this business scenario. We believe that the other business scenarios presented below can also benefit from standard QualOSS assessment method. In most cases, these other scenarios need the same kind of information and in other cases because it will be possible to adapt standard QualOSS assessment method to obtain answer to new questions.

Scenario 2: FlOSS Exploitation

This scenario corresponds to an enterprise that wants to integrate an F/OSS component in its product. Furthermore, in this scenario, the enterprise does not currently consider contributing to the corresponding F/OSS endeavor beside reporting bugs eventually.

This scenario may be imposed either because the F/OSS community producing the F/OSS component in question is not open to accepting new contributions or the most likely case, because the enterprise integrating the F/OSS component does not have the time to contribute to the F/OSS endeavor at the present time. This scenario occurs when the F/OSS component is well separated from the software product in which it will be integrated such as a F/OSS library or a F/OSS platform. In such cases, the integration happens through a well defined set of interfaces.

Although the enterprise does not consider contributing to the F/OSS endeavor producing the F/OSS components to be integrated in its product, it is still very important to have an assessment of the robustness and evolvability of the F/OSS endeavor. In particular, if the F/OSS endeavor is not robust that means, for instance, that many support questions will not receive response. If the F/OSS endeavor is not evolvable then it means that it runs the risk of becoming inactive in the longer term hence forcing the enterprise to take over the code or left to find another F/OSS endeavor producing a similar F/OSS component. In all cases, lack of robustness or evolvability in an F/OSS endeavor could incur significant cost to the enterprise.

Scenario 3: FlOSS Fork

This scenario corresponds to an enterprise that wants to integrate an F/OSS component in its product by creating a fork of the current F/OSS component. Obviously, in this scenario, the enterprise will not want to contribute to the corresponding F/OSS endeavor. Depending on how much modification are performed on the F/OSS component to integrate it in the software product, even reporting bugs to the F/OSS endeavor may not take place.

Although this particular scenario is not very *FlOSS minded,* it can definitely take place in real life since many F/OSS licenses such as BSD or Eclipse allow for the reuse of F/OSS code in proprietary applications. This scenario may also take place with code under a GPL license. However, this means that the enterprise will keep the new fork open or not distribute it. This situation may happen if the enterprise feel its contributions are not being integrated rapidly enough by the F/OSS endeavor.

In this scenario, the assessment of a complete F/OSS endeavor is not useful. Instead the enterprise will want to focus the assessment on product quality. However, we observe that the first business scenario above will also require inspecting product quality fairly intensively. In turn, the part of product assessment done by the standard QualOSS assessment method would likely provide useful result for the F/OSS fork scenario.

Scenario 4: FlOSS Takeover

This scenario has similarities to the F/OSS fork scenario except that instead of forking the project, an enterprise takes over an existing F/OSS endeavor. The important aspect of this scenario is that the enterprise usually wants a large say on how the F/OSS endeavor will be organized and controlled. Again in this scenario, the main emphasis of an assessment will be put on code quality however in some cases, the F/OSS endeavor being taken over is not necessarily inactive consequently, the enterprise may also want to consider assessing aspects related to the existing community members, the software processes and the tools and libraries used by the F/OSS endeavor.

Scenario 5: FlOSS on Infrastructure

This scenario involves the integration of F/OSS components on the infrastructure of an enterprise. However, in many cases, current ad-hoc assessment methods or other existing methodologies such as QSOS or OpenBRR already provide an adequate support to make a selection of FlOSS endeavor. The cases where the QualOSS methodology could help are those where the stake of integrating the F/OSS components in an infrastrucutre are high, for example, when the FlOSS components are complex to learn and configure and will require significant effort from the infrastructure administrators, or when many users will require training on the selected F/OSS component, or when customers external to the enterprise will have access the select F/OSS component.

When stakes are high, the enterprise definitely wants to know about the robustness and evolvability of the F/OSS endeavor producing the F/OSS component to integrate on its infrastructure. In particular, if a significant amount of resources is spent on a F/OSS component and then the corresponding F/OSS endeavor becomes inactive or even worst, no one provides support nor corrects flaws in the F/OSS component then all that resources devoted thus far on the FlOSS component could be lost. In this scenario,

an assessment would likely not focus much on code quality and would rather assess the community, its software processes and probably briefly review dependencies from the desired F/OSS component on other F/OSS components.

Scenario 6: Service on FlOSS

In this scenario, the enterprise wants sell a service on a F/OSS components. Currently, many SME sell support on renown F/OSS components. However, due to competition, the market is cut-throat. It would therefore also be useful for SME to sell service on less know F/OSS components specific to certain lucrative niche market. However, in these cases, the F/OSS components are less renown and to avoid wasting resources learning an F/OSS component with no future, it is important to assess the evolvability and even robustness of the corresponding F/OSS endeavor.

3.2.2 Introspection Scenarios

Beside the comparison scenarios where an assessment compare several FlOSS endeavor, the interview conducted during WP1 also identified that people would like guidance on how to monitor their involvement in an existing F/OSS endeavor, or how to set up a new F/OSS endeavor.

The usual goal of introspection scenarios is to provide a SWOT (Strengths, Weaknesses, Opportunities and Threats) of the F/OSS endeavor analyzed. In other words, it consists in analyzing work products, community members, tools and software processes of the selected F/OSS endeavor to identify its strengths, weaknesses, opportunities and threats with regards to the F/OSS endeavor's robustness and evolvability.

The main focus of the introspection depends on the specifics of who wants to perform the introspection. Here are example where the demand for an introspection is feasible.

- An organization considers releasing a software component under a F/OSS and they would advise on how to set up the corresponding F/OSS endeavor to increase its robustness and evolvability.
- An enterprise already produces in a F/OSS components and would like a SWOT of the corresponding F/OSS endeavor to increase its chance of success by increasing the endeavor's robustness and evolvability.
- An enterprise want to monitor more closely its involvement in an F/OSS endeavors to make sure its can establish an more accurate ROI.

WP5 will perform a case study on an introspection of the first type. In particular, two Belgian parliaments have hired a software development firm to develop applications mostly made of new Plone modules. The details of this scenario are described in deliverable D5.1.

These Plone modules will be released under a GPL-compatible license. However, merely releasing F/OSS components under a F/OSS license will not make them into a successful F/OSS endeavor. In consequence, the two Belgian parliaments want advice on how to increase their F/OSS endeavor's chance of success by setting it up so it is robust and evolvable.

4 The QualOSS Methodology

This section describes the QualOSS methodology. In Section 4.1, issues regarding the overall QualOSS methodology are presented. It mainly includes the requirements imposed by the QualOSS methodology. In addition, information related to influential research works is also presented in the first subsection.

The most important requirement of the QualOSS methodology is the prescription of the QualOSS assessment process that all QualOSS assessment method must respect. Consequently, the second subsection presents this assessment process in details.

4.1 QualOSS methodology: Requirements and Influential Works

The requirements imposed by the QualOSS methodology on assessment methods are listed in the first subsection. Furthermore, an explanation regarding how the QualOSS methodology achieves rigor and flexibility is proposed. Then, other researchers' works that have influenced the QualOSS methodology are briefly described.

4.1.1 Requirements of the QualOSS Methodology

The requirements enumerated below are mostly requirements imposed by the QualOSS methodology to assessment methods. That is, requirements that an assessment method must follow and respect in order to be called a *QualOSS assessment methods*. Only requirement 1 concerns the QualOSS methodology only.

Requirement 1. The QualOSS Methodology must be assigned a version. The QualOSS methodology described in the deliverable D4.1 submitted to the European Commission is ***Version 1*** of the QualOSS methodology.

Requirement 2. A QualOSS assessment method must have a unique name and it must have a version number. Name uniqueness may be achieved using a version number.

Requirement 3. A QualOSS assessment method must state the version of the QualOSS methodology that it follows.

Requirement 4. A QualOSS assessment method must clearly and *briefly* state the F/OSS acquisition for which it is targeted.

> A longer description of the F/OSS acquisition targeted by a QualOSS assessment method is required as part of the assessment process prescribed by the QualOSS methodology, in particular, as one objective of the task of initiating an assessment. For satisfying this requirement, it is sufficient to only provide a short paragraph description of the F/OSS acquisition where assessment results are expected to help. In other words, one should know the broad F/OSS acquisition context for which the assessment method was designed.

Requirement 5. A QualOSS assessment method must clearly state the expected effort in person-month to obtain assessment results when executing it appropriately. It is acceptable to provide a effort range and to assume familiarity of the assessment method when providing the effort range.

Requirement 6. A QualOSS assessment method must follow and respect the assessment process ***prescribed*** by the QualOSS methodology.

This assessment process is divided in a series of tasks: (1) initiating an assessment, (2) setting up and planning an assessment, (3) collecting and analyzing data, (4) interpreting results, and (5) supervising an assessment. Details of each task are given in Section 4.2.

An assessment method is said to respect the process if the assessment method is clearly partitioned into the 5 tasks of the assessment process and if it is clear that assessment results obtained using the assessment method reach the objectives set for each task of the assessment process.

Requirement 7. A QualOSS assessment method must enforce traceability during assessments and in particular, in the assessment results.

An F/OSS endeavor assessment is said to be traceable if it is possible to recall all input-output relationships including information regarding the procedures, techniques, and tools used to process an input into an output. Recording the name and version of a procedure, a technique, or a tool is a sufficient mechanism for maintaining the trace of processing.

It would be logical to anticipate that an assessment method satisfies the traceability requirement if assessment performed with the assessment method are traceable. However, this would be too strong of a burden on assessment methods. Indeed, methods may require for the recording of traces but if that recording is manual, an evaluator may ignore the requirement.

As a consequence, an assessment method is said to satisfy the traceability requirement if its implementation of the 5 tasks of the assessment process requests the recording of all input/output relationships including processing information.

The only limiting constraint is the 5 tasks imposed by the assessment process. These 5 tasks are quite generic hence the assessment process is flexible. Consequently, the QualOSS methodology is also quite flexible.

Requirement on uniqueness of names is easy to satisfy. The traceability requirement is much more tedious to meet but it does not limit the flexibility of the QualOSS methodology. Together, the requirements of uniqueness of names and of traceability impose a rigor to QualOSS assessment method. Furthermore, the activity system framework, presented in the next section, used as an underline based for specifying the assessment process also increases one's confidence in the rigor of the QualOSS methodology.

4.1.2 Influential Works

This section highlights other researchers' works that influenced the vision instilled in the QualOSS methodology, in particular, the open process framework and the activity theory.

First, we point out the work by Henderson Sellers (Henderson-Sellers, 2005, which augments the open process framework (OPF) with methods and techniques to assess commercial-of-the-shelf (COTS) components help shape the vision on for the QualOSS methodology. In his work, Henderson Sellers explains that assessment exercises are always a bit different from each other. In turn, an assessment methodology should only be based on a generic assessment process that can be adapted to various assessment situations. Similarly, we observed that there are various reasons for acquiring F/OSS components; some of these reasons were presented in the business scenarios of Section 3.2.1.2. QualOSS recognizes that each business scenario and even each instance of a business scenario could benefit from a specific, custom-fit assessment method. This was the main reason why the QualOSS methodology took the decision not to impose a single assessment method but rather to prescribe only a high-level assessment process. However, from a pragmatic standpoint, it is not cost effective to design a new assessment method for every case of F/OSS acquisition encountered. Consequently, Section 5 proposes the standard QualOSS assessment method. This standard assessment method was created with a particular business scenario in

mind however, that scenarios addresses many questions raised in other F/OSS acquisition scenarios. Consequently, we believe that even the standard QualOSS assessment methodology is quite flexible.

A second research area that influenced the QualOSS methodology is related to activity theory (Bjorke, 2005, de Souza, 2003). Initially, the activity system framework of activity theory, shown in Figure 3, was used to develop our definition of F/OSS endeavor presented in Section 2. In particular, the tuple of 4 sets (WP, CM, SP, TL) that define an FlOSS endeavor relates as follows to the elements of the activity system framework shown in Figure 3. WP corresponds to the object of activity system, CM relates to subject and community, SP aggregates rules and division of labor, and TL is a direct mapping to tools.

Incidentally, it is worth noting that OPF defines the term *endeavor* in a similar way to our definition of F/OSS endeavor, the difference being that its definition fits the traditional view of an enterprise endeavor while ours is adapted to the F/OSS context.

Activity theory also provides a comprehensive framework on which we can rely to craft the assessment process prescribed by the QualOSS methodology. First, we quickly explain the activity system framework illustrated in Figure 3. In particular, it identifies the elements interacting with each other when an human activity is performed. In more details, activity theory defines an activity as a conscious action performed by the subject on an object in order to achieve a motivated outcome. Besides, the subject uses tools and interacts with a community to perform the activity. In turn, there is an implicit or explicit division of labor between the subject and the rest of the community. Furthermore, the activity is performed in a particular context where certain rules must be respected. Consequently, all human activity including an F/OSS endeavor assessment, take place in the activity system framework above.

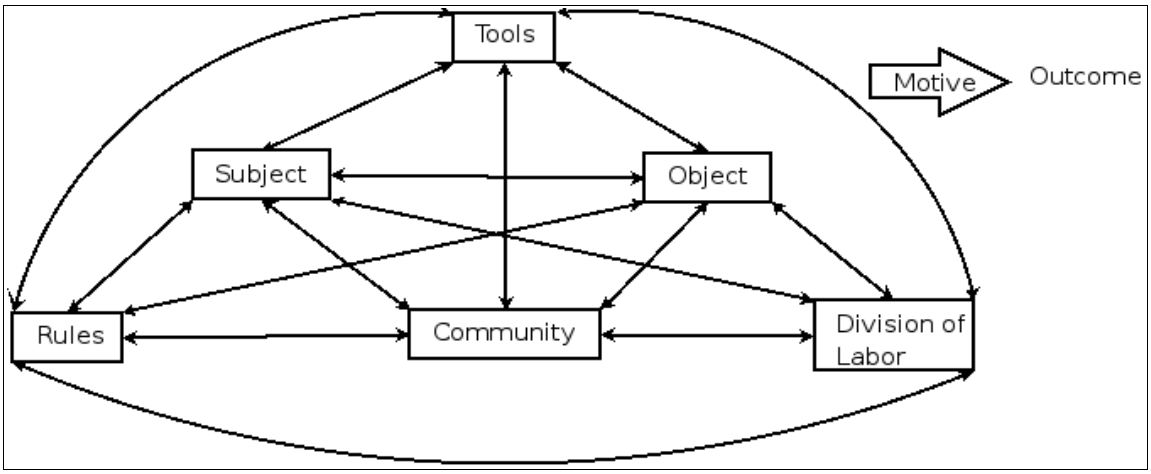

Figure 3 - An activity system: its elements and their interactions when an activity takes place.

Relating Figure 3 to the activity of assessing an FlOSS endeavor gives the following: The subject is a person involved in a F/OSS assessment, for instance, an evaluator; the object is a F/OSS endeavor whose assessment has been motivated by the need to obtain an expected outcome. The tools are the various methods and techniques used by evaluators when conducting the assessment activity. The community includes other people involved in an F/OSS assessment, for example, other evaluators or members of the F/OSS community who will be asked to validate assessment results. The division of labor is the implicit or explicit partition of effort needed to produce the results of an assessment. Finally, the rules are the principles and conventions established by the QualOSS methodology in general as well as those specific to the assessment of an given F/OSS endeavor.

To build a rigorous assessment process using the activity system framework, we can check that the 5 tasks of the assessment process cover all the elements of an activity system and their interactions. Consequently, in the subsection of Section 4.2, each task of the assessment process clearly explains what elements of an activity system are covered as well as what relationships between elements are concerned.

Adding the coverage of the activity system framework to the requirements in the previous section is yet another step that increases one's confidence in the rigor of the QualOSS methodology.

4.2 Assessment Process of the QualOSS Methodology

This section presents the the assessment process prescribed by the QualOSS methodology. This assessment process is divided in a series of 5 tasks: (1) initiating an assessment, (2) setting up and planning an assessment, (3) collecting and analyzing data, (4) interpreting results, and (5) supervising an assessment.

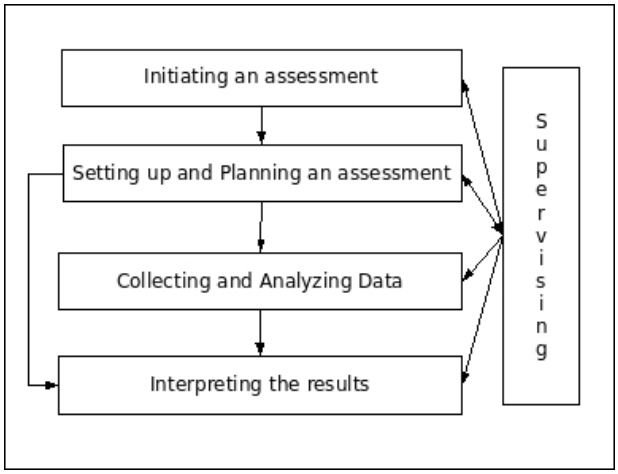

Figure 4: Dependencies between tasks of the assessment process.

As illustrated Figure 4, the first 4 tasks are performed in sequence while the last task of supervision is transversal and executed throughout the assessment process. The arrows show the input/output dependencies between tasks.

Each task of the assessment process is presented in its own subsection below. Each tasks lists its objectives and its outcomes. Furthermore, it specifically highlights the connection with the part of the activity system framework covered. Finally, each task also mentions the procedures, tools, and techniques expected to help to conduct each task, that is, the kind of procedures, tools, and techniques that one may expect to use when building a QualOSS assessment method.

4.2.1 Initiating an Assessment

This section presents the objectives of the task *initiating an assessment* and relates these objectives to the activity system diagram show in Figure 3. It then lists the expected outcome from the task. Finally, it describes the kind of procedures and techniques likely to be useful for conducting the task *initiating an assessment* efficiently.

The task of *initiating an assessment* has ***three objectives***:

- Describe the broader context in which the assessment will take place, (e.g. the software development project that plans to integrate the F/OSS component)

- State why an assessment is needed (in relation to a business viewpoint),
- Explain how the outcome of an assessment will be used in the given broader context
- Enumerate business constraints that the assessment activity must respect, e.g., limits on cost, timing and effort. These business constrains should derive from the first two objectives above.
- Validate the outcomes produced by the objectives above.

Phrasing the objectives above relative to the activity system diagram of Figure 3, we find that the first two objectives state the ***Motive*** for an assessment and the third objective describes the expected ***Outcome***. Furthermore, by describing the broader context in which an assessment takes place, this task describes how this overall assessment activity fits with or connects with other related activities of the larger project, i.e., activities that take the results of an assessment activity as input. This broad description therefore substantiates how to structure the outcome of an assessment so other activities can use it effectively. Finally, the business constraints identified by the last objective represent ***rules*** in Figure 3.

The ***expected results*** related to these objectives are:

- A statement presenting the business viewpoint of why the assessment is useful

  As a result of this point, it should become clear which usage scenario between comparison or introspection presented in Section 3 applies in the given situation.

- A description of the broader context in which an assessment will take place

  This statement should describe in enough details the complete situation in which the assessment takes place. In other words, this description should explain how the results of an assessment activity will be used. It may be useful to present the complete project during which the assessment activity takes place.

- A list of information/data expected to result from an assessment

  This list will help decide on how to structure the result(s) of an assessment so they are presented efficiently to other activities of the broader project.

- A list of business constraints that must be respected by the assessment activity

  This list for example mentions the maximum person-day effort to spend on the assessment activity. A timing constraints may also be added, such as, the results must be available in 1 week. If several parties are involved in the assessment activities, it is probably important to have everyone's agreement or to ask each party to provide their own business constraints in order to compile the global assessment constraints from those given by each party. At this stage, not every details of an assessment are known, the business constraints enumerated are derived from top down in turn they set the bounds for the 4 remaining tasks.

Furthermore it is expected that the outcomes listed below are valid. In other words, they must directly come from a customer. However, if a particular application of the QualOSS methodology suggests creating a standard assessment activity for a recurring scenario, then it is expected that the outcomes resulting from the task initiating an assessment are validated before continuing with the next task.

Below we enumerate procedures and techniques that will likely assist in conducting the task *Initiating an Assessment.* These procedures and techniques must ensure that the objectives of initiating an assessment are reached and also that the outcomes are structured appropriately to continue with the next task, namely, *setting up and planning the assessment*.

Consequently, the ***procedures and techniques*** to help with the task *Initiating an Assessment* are:

- Standard questionnaires that help to identify the context and the business needs behind an assessment. These standard questionnaires will then be used to interview various stakeholder in the project requiring an assessment. To guarantee that the objectives of the task *initiating an assessment* are fulfilled, the standard questionnaire should contain questions that address the 4 objectives.
- Lists of past contexts where assessments based on the QualOSS methodology were conducted and provided useful results
- Interview methods that help stakeholders to identify the business needs where assessments based on the QualOSS methodology are needed. Interview methods should be based on the standard questionnaires mentioned in the previous point. In addition, these methods should augment questionnaire to ensure that the results from interviews provide the needed information and that this information was obtained in an efficient way. For example, the method may help the interviewer to identify the appropriate stakeholders to interview or to identify the relevant assessment contexts that speak to stakeholders and where the QualOSS methodology delivered useful results
- Document templates for presenting effectively the results of the task *initiating an assessment*. This will ensure that all the information expected as outcome of this task are provided and no important information or data is involuntarily omitted.

4.2.2 Setting up and Planning an Assessment

In this section, we describes the task of *setting up and planning an assessment*. It uses the results provided by the task *initiating an assessment* to set up and plan how to conduct the remaining of the assessment. This section first presents the objectives of the task *setting up and planning an assessment*. It then relates these objectives to the activity system diagram of Figure 3. It also specifies the outcomes expected of this task and then describes the procedures and techniques likely to be useful for conducting this task.

The ***objectives*** of *setting up an assessment* are to:

- Identify the accurate F/OSS endeavor(s) to assess, that is, the FlOSS endeavor's scope must be specified clearly and accurately
- Select the people who will perform the assessments and also identify the broader community of people likely to be impacted by the assessment (either because they will be contacted and asked to participate in the assessment activity, or because they may be impacted by the assessment results)
- Specify how people participating in the assessment will share the workload (using a workflow for example) and determine the rules that they will obey when performing the assessment.
- Identify the tools to use during the assessment activity and how these tools must be tailored or configured for this particular assessment. We note that in this context, the term *tools* is to be taken broadly, that is, tools may be automated (e.g. software tools), manual (e.g., manual procedures) or partially automated. For instance, manual procedures to follow to validate data or to evaluate documentation are considered tools and thus must be clearly identified. Naturally, software analysis tools should also be identified and included in the tools set to use later in the task of the assessment.
- Plan the supervision strategy for the remaining part of the assessment and communicated to the person in charge of the supervision task.
- Validate the outcomes produced by the objectives above

Regarding to the activity system diagram of Figure 3, the task of *setting up an assessment* corresponds to identifying the ***object*** (F/OSS endeavor(s) to assess), the ***subject*** (people who perform the assessment) and the ***community*** (people impacted by the assessment), the ***division of labor*** (sharing the workload) and the ***rules*** (rules obeyed during an assessment). Finally, the last objective corresponds to identifying and setting up the ***tools*** for the remaining of the assessment activity. It is worth noting that tools are needed to collect and analyze data, but the guidelines for interpreting the results and the methods and techniques to use to supervise an assessment activity are also considered tools. In turn, the subtask of identifying (or selecting) tools is a complex one.

Although there is not necessarily a strict order in which to attain the objectives, the usual order is given in the list above. In particular the first step consists of identifying the F/OSS endeavors to assess then the people who will perform the assessment. If specific people cannot be identified then the roles and expertises required to conduct the assessment activity should be stated.

The subsequent step is to identify the broader community of people impacted by an assessment, for instance, people from the F/OSS endeavors being assessed whose assistance will be requested for validating data and data analysis results.

The next step consists of identifying the rules to obey during the assessment as well as how the workload will be divided among the people performing the assessment. Clearly, the QualOSS assessment method used by evaluators already imposes certain rules when performing an assessment but additional rules will likely be useful for each different F/OSS endeavor and specific assessment context.

Finally, the bulk of setting up an assessment is to identify and set up the tools for the remaining of the assessment task. This consists in setting up the software analysis tools but also all the other procedures and techniques needed for conducting an assessment activity such as a procedure to customize a quality model or indicators, techniques for validating procedures and guidelines to follow during an assessment, etc. This step is quite complex because many kinds of tools must be identified and in many cases, these tools will require tailoring to be fit for a given assessment.

Below, we identify the ***expected results*** from the task of setting up an assessment. These results are derived from the objectives listed above.

- A list of F/OSS endeavors to assess. To perform a reliable and valid assessment, the scope of the F/OSS endeavor must be specified accurately; for example, are we assessing a set of projects or a part of a F/OSS project?.
- A list of expertises needed to perform the tasks required by this assessment activity.
- A list of people or roles of people who will conduct the assessment
- A list of people potentially involved in the assessment and eventually people potentially impacted by an assessment
- A list of rules to obey during the assessment activity. These rules will augment the business constraints already identified in the previous task when initiating the assessment.
- The list of tools that have eventually to be tailored or configured to help in the execution of the remaining tasks of the assessment activity. We emphasize again that in this context, tools mean any methods or techniques used in an assessment activity including software tools but also workflows, guidelines, validation techniques, etc.
- Workflows to follow in the remaining tasks of the assessment activities, in particular, the tasks of collecting and analyzing data, interpreting results, and supervising the assessment. These workflows should be as

precise as possible in identifying who performs what actions using which tools at what moment. Clearly, it is not always possible to give a precise workflow at the setting up stage however, it should normally be possible to give an initial workflow with missing information that will be completed when available in later tasks of the assessment activity.

The list below presents the ***procedures and techniques*** helpful for conducting the task of *setting up an assessment* efficiently. The ease of a set-up and planning task is closely related to existing past experience. In particular, if a F/OSS acquisition context to assess is similar to a past assessment context then we can leverage on the QualOSS assessment method used in that past assessment and only perform little customization to it for this new assessment. In turn, the most important technique to have in place to ease setup and planning is a repository to collect the experience from each assessment performed so future assessments can refer to these past experiences.

- Guidelines and workflow to follow when conducting the task *setting up an assessment* (These guidelines and workflows could be based on the series of steps mentioned above in our textual description of how to conduct the setting up of an assessment.)
- List of expertises that people need to conduct the various actions of past workflows.
- Template to specify systematically the various F/OSS data sources that contain needed data to evaluate a F/OSS endeavor.
- Set up a repository of various workflows used in past assessments activities for conducting the tasks of collecting and analyzing data, of interpreting the results and of supervising an assessment.
- Past techniques used to define the scope of an F/OSS endeavor accurately
- Set up a repository of methods used in the past for evaluating the quality of data available for F/OSS endeavors
- Set up a list of people roles that were impacted by or involved in past assessments
- Set up a list of rules that an assessment must often respect, including list of risks to manage during an assessment
- Set up a repository of methods and tools that have been used in the past for collecting and analyzing data, for interpreting results or for supervising an assessment along with the various configurations used for each tool
- Set up a repository of techniques used in the past for validating the methods and tools used during an assessment.

4.2.3 Collecting and Analyzing Data

This section presents the objectives of the task *collecting and analyzing data* and relates these objectives to the activity system diagram show in Figure 3. It then lists the expected outcomes from the task. Finally, it describes the kind of procedures and techniques likely to be useful for collecting and analyzing data efficiently.

We note that the procedures and tools for collecting and analyzing data were already selected and eventually customized during the task of setting up the assessment (Section 4.3). Therefore, this task 's job should be limited to executing the selected procedures and tools to obtained analyzed data. In many cases however the procedures and tools selected and set up during the setting up task will run into problems when

actually collecting and analyzing data. It is therefore the job of this task to adjust these procedures and tools to work properly on the particular F/OSS endeavor under assessment.

The task of collecting and analyzing data has the ***four main objectives***:

- Collect the data whose sources were identified when setting up the assessment
- Analyze the data using the methods selected when setting up the assessment
- Maintain a link between the raw data, the methods and tools used to collect and analyze that data, the results of data analysis.
- Validate the collected and analyzed data

When all happens as planned, the first two objectives are easily achieved by executing the methods selected when setting up the assessment during the previous task. However, in many cases, new issues arise only when actually collecting and analyzing data. In such cases, the methods and tools selected during the set up task must be adapted in order to collect and analyze data appropriately. It is therefore the responsibility of this task to adapt these methods and tools so they work properly. Hence, the ***three sub-objectives*** below must often also be satisfied:

- (If needed) Adapt the select methods and tools to collect and analyze data properly.
- (If needed) Validate the adapted methods and tools before collecting and analyzing data
- (If needed) Record how methods and tools were adapted. (It may eventually be useful to create a whole new version of the adapted method or tool.)

Phrasing the objectives above relative to the activity system diagram of Figure 3, we find that the first two objectives make the evaluators (***subjects***) use methods (***tools***) to collect and analyze data of an F/OSS endeavor (***object***). The third objective, which record links between raw, analyzed data and analysis methods, consists in creating additional data related to the ***object*** (the F/OSS endeavor assessed). The first sub-objectives relates to ***tools*** since it consists in adapting methods (***tools***). The other two sub-objectives impose to validate and record how methods are adapted hence they create new ***rules*** for the assessment activity.

The ***expected outcomes*** resulting from reaching these objectives are:

- A set of measures taken on the F/OSS endeavor(s) to assess.
- Links between each measure taken, the raw data on which the measurement was taken and the tool(s) and the tool configurations used for taking the measurement.
- (If needed) Comments detailing how methods and tools where adapted to collect and analyze data properly.
- (If needed) Validation results of the adapted methods and tools.

It is worth mentioning that the evaluators collecting and analyzing data are supposed to follow the workflow defined during the assessment set up in the previous task. However, it is the role of the task of *supervising an assessment* to verify that the workflow is respected and that each operation of the workflow operation is executed appropriately.

Below, we list the kind of methods and techniques expected to help execute to the task of c*ollecting and analyzing data* and guarantee that it achieves the task objectives and sub-objectives.

Furthermore, some of the procedures and techniques listed may also assist in presenting the task results appropriately to the next task of *interpreting the results*.

Consequently, the ***procedures and techniques*** to help with the task Collecting and Analyzing Data are:

- Database schema with integrity constraints requiring links between raw data sources, the analysis tools and their configurations, and the results obtained from analyses
- A software platform that automates the collection and analysis of data while maintaining the required links between raw data, analysis methods and tools and the results of analysis.
- Guideline on how to adapt the software platform to new assessment contexts
- Template listing the type of information to provide to validate a tool so it may be connected to the QualOSS platform (including information related to the various configurations used for the tool)
- Guideline on how to validate a new data analysis tool so it is acceptable to plug it in the QualOSS platform
- Manual procedures that explains how to collect and analyze data and how to insert them in a database that respect the given schema. (This kind of procedure is expected to be useful when a software platform is not available or for cases where a software platform is not available)
- Methods to validate adaptations to methods and tools used for collecting and analyzing data
- Template of document for collecting customizations performed to methods and tools and results of validating these customizations
- Methods for evaluating the quality of data available for a particular F/OSS endeavor (this can be done on the raw data or on the analyzed data)

4.2.4 Interpreting the Results

This section presents the objectives of the task *interpreting the results*. It shows how this task relates to the activity system diagram of Figure 3. Finally, it describes the kind of procedures and techniques expected to help in interpreting results.

The task of collecting and analyzing data has the ***three objectives***:

- Interpret the analyzed data resulting from the task *collect and analyze data* (Section 4.4) using the interpretation methods selected during the task of *setting up and planning an assessment* (Section 4.3).
- Maintain a link between the analyzed data, the interpretation methods, and the resulting interpretation.
- Validate the resulting interpretation

When all happens as planned, it is then possible to fulfill the three objectives. However, if issues were encountered when collecting and analyzing data then they may have impacted how the analyzed data should be interpreted. It is therefore the responsibility of this task to adapt the interpretation methods. Hence, the sub-objectives may also need to be satisfied:

- (If needed) Adapt interpretation methods so they interpret the analyzed data appropriately.
- (If needed) Validate how the interpretation methods were adapted.
- (If needed) Record how the interpretation methods were adapted.

Phrasing the objectives above relative to the activity system diagram of Figure 3, we find that the first two objectives make the evaluators (***subjects***) use methods (***tools***) to interpret analyzed data of an F/OSS endeavor (***object***). The third objective which requires recording links between analyzed data, interpretation methods and resulting interpretation consists in specifying a ***rule*** that the assessment activity must respect. The first sub-objective relates to ***tools*** since it consists in adapting interpretation methods (***tools***). The next two sub-objectives imposes the new ***rules*** to the assessment activity by requiring to validate and record how interpretation methods are adapted.

The ***expected outcomes*** resulting from reaching these objectives are:

- A set of indicators that allow assessing the robustness and evolvability of the selected F/OSS endeavors.
- Links between each indicator, the analyzed data from which the indicator was derived, and the interpretation methods used to derive each indicator.
- (If needed) Comments on how interpretation methods were adapted.
- (If needed) Validation results of the adapted methods.

Below, we list the kind of ***procedures and techniques*** expected to help to execute the task of interpreting results (analyzed data).

- Quality models for adjust the meaning of robustness and evolvability to a particular assessment context. Quality models may be constructed using the Goal-Question-Metric paradigm and eventually presented as a tree hierarchy of goals and then questions to answer (based on metrics) to determine if a goal is fulfilled. Furthermore, the goals and questions can also be weighted.
- Techniques to determine how to weight each item in a quality model to reflect the assessment context correctly.
- Template for presenting the results of an assessment.
- Techniques to present assessment results so that others may comment, for example, to allow a F/OSS community member to comment certain results of an assessment or to allow an other evaluator to question certain results, etc.
- Guideline to produce a new quality model or tailor an existing one and validate it with regards to the broader assessment goals.
- Guideline to validate a new indicator that aggregates metrics and propose thresholds on this aggregation in order to answer questions raised (or characteristics present) in a quality model
- Techniques to validate interpretation methods, e.g., guideline for validating a proposed indicator
- Techniques for validating small adjustments to interpretation methods
- Template of document for collecting adjustments performed on interpretation methods

4.2.5 Supervising an Assessment

This section describes the task *supervising an assessment.* First, it presents the task objectives and its expected outcome. It also shows how this task relates to the activity system diagram of Figure 3. Finally, it lists the kind of methods expected to help in interpreting results.

It is important to note that the supervision of an assessment is a transversal task since it runs from the beginning to the end of an assessment activity.

The task of *supervising an assessment* has the following ***two objectives***:

- Monitor that all the tasks of the assessment process are executed as planned including that each task performs the validation of its outcome.

  For example, if a workflow is given for the execution of a task then it is the responsibility of the supervision task to monitor that the work flow is respected. Furthermore, it is also the role of the supervision task to verify that the operations of a workflow are performed appropriately. For example, if the workflow of the task *collecting and analyzing data* states that raw F/OSS data must be validated by interviewing three F/OSS community members then the supervision task must verify that three F/OSS community members agreed to review the selected datasets and then validate them. In practice, this may simply be done by maintaining a document that logs the operation to supervise and then verify after the fact that these operation were done.

- Record the tension that occurred during the whole assessment activity

  At a higher level, it is also interesting to monitor how an assessment activity occurred to improve later occurrences of assessment activities or at least to indicate where to focus the supervising task. For example, if we notice that evaluators never recorded various tool configurations used to analyzed data. It may then be appropriate to conduct a short training on this particular aspect so that they will perform such recording in later assessment activities.

Phrasing the objectives above relative to the activity system diagram of Figure 3, we find that the first objective verify that the ***rules*** and ***division of labor*** specified in workflows of each task are executed as prescribed. The second objective works at a high level and observes the whole assessment activity. In particular, it observes the interactions between the six elements of the activity system (***subject, object, tools, community, rules, and division of labor***) to identify eventual tension points between two or more elements). This higher level perspective is not useful to solve issues for future assessment activities.

The ***expected outcomes*** resulting from reaching these objectives are:

- List of operations and operation sequences that were supervised with the information on whether it took place appropriately.
- Comments collected from the various people involved in the assessment activity.
- Comments extracted from reviewing of the outcomes produced by the other 4 tasks
- List of issues identified for the current assessment activity and action plan on how these issues will be handled for later assessment activities.

Below, we list the kind of ***methods and techniques*** expected to help execute to the task of supervising an assessment.

- Template for specifying operations and operation sequences to supervise. It may be useful to request that timestamps be given for the beginning and end of each operation.
- Template for evaluators to provide their comment in a semi-structure way.
- Guideline on how to review the outcomes from the other 4 tasks.

- Template to facilitate the collect of the appropriate information when reviewing the outcome from the other 4 tasks.
- Questionnaires to collect comments from the various people involved in an assessment activity.
- Template to list the various issues identified from the outcome of the current assessment
- Repository of issues identified in past assessment and actions taken to eliminate, to avoid them or to reduce their impact on the assessment activity.

5 Assessment Methods Based on the QualOSS Methodologies

This section mainly describes an assessment method that follows the QualOSS methodology and fulfills all of the QualOSS methodology requirements presented in Section 4. This assessment method comprehensively described in Section 5.1 is considered the standard QualOSS assessment method. It is also referred to as the middleweight assessment method in relation to the effort needed to obtain the results of the assessment for an F/OSS endeavor fixed in the range of 1 person-week to 1 person-month.

In addition, lightweight and heavyweight assessment methods are also proposed in Section 5.2 and 5.3 respectively. Briefly stated, the lightweight method consists in exploiting assessment results already obtained using the middleweight assessment method. To achieve this, the approach proposed is to create a shared repository where all the results of F/OSS endeavor assessments may be uploaded and shared. The added benefits of sharing results is open validation and open debate on assessment results. It is nonetheless important to emphasize that the lightweight assessment method goes beyond merely sharing assessment results, it is a complete QualOSS assessment method in its own right. It satisfies all the requirements of the QualOSS methodology.

In Section 5.3, guidelines on how to conduct an heavyweight assessment are presented. The heavyweight approach can be viewed as an advanced customization of the middleweight assessment method proposed in Section 5.1. There are many ways to customize the middleweight assessment method, in consequence, many heavyweight methods could results. The goal of Section 5.3 is not to describe a single heavyweight method but rather to highlight the part of the middleweight assessment methods that can be customized and to propose guidelines to perform the customizations.

5.1 Standard QualOSS Assessment Method : *Assessment for Full FLOSS Collaboration*

The standard QualOSS assessment method described in this section fulfills all the requirements set by the QualOSS methodology. In Section 5.1.1, each requirement of the QualOSS methodology is reiterated and then a satisfaction clauses is given on how the standard QualOSS assessment method fulfills that requirement. Section 5.1.2 then presents how the standard QualOSS assessment method implements the assessment process prescribed by the QualOSS methodology.

5.1.1 Satisfying the QualOSS methodology requirements

Requirement 1. The QualOSS Methodology must be assigned a version. The QualOSS methodology described in the deliverable D4.1 is currently ***Version 1*** of the QualOSS methodology.

> ***Satisfaction Clause for Requirement 1:***
>
> requirement related to the methodology itself, not applicable for assessment method.

Requirement 2. A QualOSS assessment method must have an unique name and it must have a version number. Uniqueness may be achieved using a version number.

> ***Satisfaction Clause for Requirement 2.***
>
> This middleweight assessment method is named: ***Assessment for Full FlOSS Collaboration – Version 1***[3]

[3] The QualOSS assessment method named ***Assessment for Full FlOSS Collaboration – Version 1*** is also referred to as the middleweight assessment method and it is also considered to be the standard QualOSS assessment method. Thus, they are synonyms.

Requirement 3. A QualOSS assessment method must state the version of the QualOSS methodology that it follows.

> ***Satisfaction Clause for Requirement 3.***
>
> A QualOSS assessment method Assessment for Full FlOSS Collaboration – Version 1 follows ***version 1 of the QualOSS methodology***.

Requirement 4. A QualOSS assessment method must clearly and *briefly* state the F/OSS acquisition for which it is targeted.

> ***Satisfaction Clause for Requirement 4.***
>
> *Assessment for Full FlOSS Collaboration – version 1* targets a F/OSS acquisition scenario where an enterprise plans to integrate a F/OSS component in a software product. Furthermore, it intends on fully collaborating with the existing F/OSS endeavor whose component is integrated. This full collaboration goes both ways. That is, the enterprise wants to contribute to the F/OSS endeavor while benefiting from others' contribution. The contributions given or received potentially cover a wide spectrum such as contribution to source code (corrections or enhancements), to documentation, to tests, or even translation.

Requirement 5. A QualOSS assessment method must clearly state the expected effort in person-month to obtain assessment results when executing it appropriately. It is acceptable to provide a effort range and to assume familiarity of the assessment method in order to perform an assessment within the effort range.

> ***Satisfaction Clause for Requirement 5.***
>
> The effort to assess one F/OSS endeavor is anticipated ***between 1 person-week and 1 person-month*** depending on the level of familiarity with Assessment for Full FlOSS Collaboration – Version 1. It expected that accustomed evaluators will on average 2 person-weeks to assess an F/OSS endeavor. This effort range is inline with the current effort invested by F/OSS integrators when they perform ad-hoc assessment of an F/OSS endeavor. This information on assessment time was collected informally during interviews conducted in WP1.

Requirement 6. A QualOSS assessment method must follow and respect the assessment process ***prescribed*** by the QualOSS methodology.

> ***Satisfaction Clause for Requirement 6.***
>
> *Assessment for Full FlOSS Collaboration – version 1* described in Section 5.1 implements the 5 tasks of the assessment process prescribed by the QualOSS methodology. Furthermore, from Section 5.1 it should be very clear that each task objective will be reached if an evaluator follows the method *Assessment for Full FlOSS Collaboration – version 1* properly.

Requirement 7. The QualOSS methodology imposes traceability to F/OSS endeavor assessments. That is, a QualOSS assessment method must require the recording of traces (between input, output, and processing) when performing the assessment of an F/OSS endeavor.

> ***Satisfaction Clause for Requirement 7.***
>
> *Assessment for Full FlOSS Collaboration – version 1* requires the recording of traces for all 5 tasks of the assessment process. Below, we briefly explain how each task requires trace recording. Additional details are found in Section 5.1.

The task of initiating an assessment does not require any trace recording.

The setting-up and planing task forces an assessment activity to explicitly state 1. the scope of a FlOSS endeavor, the evaluators and other people involved in an assessment, the rules that the assessment obliged to, and the workflow of operations (manual procedures as well as software tools) used to process every input set into final or intermediate output.

The task of collecting and analyzing data will be orchestrated by the QualOSS platform. One can observe that the database schema on the top of which is based the QualOSS platform will enforce the traceability requirement. Accordingly all traces between input, output and processing will be stored by the QualOSS platform. Hence, the database behind the platform serves as the trace repository for the F/OSS endeavor assessments performed. For a strength traceability, the QualOSS platform also requires that configuration data used by analysis tools also be recorded and that eventual trace logs generated by analysis tools also be recorded.

The task of interpreting the results is based on a quality model whose leaves are questions. In turn questions are answered using indicators whose values are obtained by aggregating the analyzed data resulting form the previous task of data collection and analysis. The QualOSS platform also compute indicators automatically and can eventually aggregate several indicator values from the leaves to the root of the quality model. The quality model including its questions, all indicators, and measures aggregated in indicators are all provided to the QualOSS platform in configuration files. In turn, a link between the configuration files used for an F/OSS assessment and the assessment results i.e., the assessment report will also be maintained through the QualOSS platform database.

With regards to traceability, the supervision task performed as part of the middleweight assessment method is used as a guard to verify that traceability is maintained in each task and throughout the assessment activity.

Based on the explanation given above as well as the complete specification of how the middleweight assessment method implements the assessment process given in the remaining of Section 5.1, we can claim that the middleweight assessment method impose traceability hence satisfy the traceability requirement of the QualOSS methodology.

5.1.2 Implementing Assessment Process for the Standard QualOSS Assessment Method

This section describes the implementation of the assessment process for the standard QualOSS assessment method called ***Assessment for Full FlOSS Collaboration – Version 1***. In particular, the implementation of each task of the assessment process for the method *Assessment for Full FlOSS Collaboration – Version 1* is described in its own subsections (5.1.2.1–5.1.2.5).

Prior, it is worth mentioning where assessment results will be collected during the execution of the QualOSS assessment method. In particular, the QualOSS methodology does not prescribe a format for assessment results. As a consequence, each assessment method is free to propose its own format for presenting the outcomes generated by the various tasks.

This middleweight assessment method proposes to capture all the information relevant to an assessment including traceability information in a single assessment report. A template of information that may be pertinent to include in this report is given in Annex B. Through the assessment process, two documents are created: first the assessment report and second the supervision report. However, at the end, the supervision report will be appended as an annex of the assessment report.

The information below is already included in that assessment report template in Annex B.

A unique name for the assessment method: ***Assessment for Full FlOSS Collaboration***

A version number for the assessment method: ***1.0***

Time to apply the assessment method: ***Between 1 person-week and 1 person-month***

Applicable to the FlOSS Acquisition Context: ***Full FlOSS Collaboration Scenario***

5.1.2.1. Initiating An Assessment for a Full FLOSS Collaboration

The objectives of initiating an assessment are first to describe the broad context of the assessment, second, to justify the need for an assessment, third, to explain how the result of an assessment will be used, and fourth, to identify high-level business constraints such as the time-frame in which results must be produced and what maximal effort or cost is acceptable.

To facilitate the execution of the standard QualOSS assessment method, we provide a generic text that satisfy these 4 objectives. That text is also already included in the template in Annex B used to create assessment reports. The text is generic since it neither mentions a particular F/OSS component nor the scope of F/OSS endeavor. Indeed, we wanted for the standard QualOSS assessment method to be applicable to all types of F/OSS component as well as to all scopes of F/OSS endeavor (mentioned in Section 3.2.1.1). The only constraint on the standard QualOSS assessment method regards the F/OSS acquisition scenario use for elaborating the method: Full F/OSS collaboration acquisition context.

Broad Context Surrounding the Assessment

In the scenario Full F/OSS Collaboration envisaged, the assessment usually takes place during a software development project. In particular, an organization wants to develop a software product, either for internal use or external distribution. An initial high-level analysis identified the broad functionality to implement in the new software product. Furthermore, to help in the implementation of the desired application, developers plan on leveraging on existing F/OSS components. They then performed a search for F/OSS components that meet some of the functional needs of the new software product. A list of F/OSS components candidates for integration has been identified.

Results of this high-level analysis were presented to the organization's management and it agreed that the proposed new software product has a good potential. However, before launching the project for the new software product, management now asks their technical staff to perform a more detailed analysis for each F/OSS candidate identified. Based on the results of this detailed analysis, the project planning will become more accurate and management will better know how to estimate more accurately a return on investment for the new software product based on the more realistic development plan.

At this point in time, each F/OSS component in the candidate list is considered one at a time. The context of this assessment correspond to the assessment of a single F/OSS component from the list. Furthermore, this assessment is specifically suited for a F/OSS acquisition scenario asking for a full F/OSS collaboration. This means a collaboration that goes both ways. That is, the enterprise wants to have the opportunities to contribute to the F/OSS endeavor; mostly to ensure that the new software product will remain compatible with the F/OSS component and integration issues between its software product and the F/OSS component in subsequent versions will be minimized. Conversely, the enterprise also wants to benefit from others' contributions, if desired. The type of contributions envisaged (given or received) covers a wide spectrum such as contribution to source code (corrections or enhancements), to documentation, to tests, or even translation work.

To decide whether the F/OSS component is worth integrating, the organization not only wants to determine the quality of the F/OSS component itself but it also wants to know about the context surrounding the F/OSS component. In particular, it is important for the organization to have information about the F/OSS community and its members as well as how appropriate their working methods are to develop and to support the F/OSS component. Furthermore, dependencies between the F/OSS components and other F/OSS libraries or between the F/OSS community and support tools should not be neglected as they provide relevant information to decide whether to acquire the F/OSS component or not.

The aggregation of the F/OSS component with its surrounding context, that is, its community members, its software processes and its tools and dependences is referred to as an F/OSS endeavor. This assessment therefore consists in identifying and evaluating the risks related to a F/OSS endeavor with regards to integrating its F/OSS component in a software product. Two important characteristics that bear on the risks of integration are 1. the capability of the F/OSS endeavor to solve current problems, and 2. the capability of the FlOSS endeavor to remain viable in the long term. This two capabilities are respectively called the robustness and the evolvability of an F/OSS endeavor. As a consequence, this assessment results present the risks related to the robustness and the evolvability of the F/OSS endeavor under assessment.

The important point addressed above is the description of an imaginary scenario where the standard QualOSS assessment method would be used.

Justification for the FlOSS assessment

F/OSS endeavors come in many shapes. Some are display a robust and evolvable behavior and provide great quality F/OSS components while others are quite weak. Therefore, when having to decide whether to integrate an exiting F/OSS component or to develop the desired component using the enterprise's resources (the make or build decision), managers want convincing indicators that the development project that plans to integrate a F/OSS component is well planned. A significant part of respecting planing will depend on the F/OSS endeavor's robustness, that is, its capability to solve current problem.

Furthermore, for high-level management to estimate any kind of return on investment, it needs indication as to the long term viability of the F/OSS endeavor, that is, its evolvability. Incidentally, the management also wants to know whether its longer term vision is shared by the other community members so that the organization's contributions will be allowed and integrated in the F/OSS component's base line.

The organization's technical staff including developers, analysts and project managers, who will be implementing the new application and integrate the F/OSS component must also be convinced of the F/OSS endeavors quality. In particular, when they create an initial planning of the development project, that includes the F/OSS integration phase. In order to established reasonable estimate, it is important for the technical staff to know about the seriousness of the support provided by the F/OSS endeavor, for instance, how long does it commonly take to obtain acceptable answers to technical issues? Furthermore, developer largely depends on the documentation such as API, code documentation and others, to integrate the F/OSS component properly in their software product. Consequently, the assessment will cover that aspect as well.

When providing or retrieving contributions to or from the F/OSS endeavor, the organization's developers will interface with the F/OSS endeavor, in particular, they will interact with other community members but they will also have to be accustom to the F/OSS endeavor's software processes and tools such as a specific version control system and a bug tracking system. Based on the ease with which the organization's procedure and tools can interface with those of the the F/OSS endeavor, providing and retrieving contributions may be facilitated or hampered. In turn, this may have an impact on the software project plan for the new application.

All the reasons mentioned above definitely show a need for an holistic assessment of an F/OSS endeavor and not merely an limit assessment of the code only. Indeed some factors related to software processes, tools, dependencies on other F/OSS components, the community and its members could weigh as much as the F/OSS component itself in the final decision to integrate the F/OSS component or not.

Use of the Assessment Results

The result of the assessment of an F/OSS endeavor will be presented to a large panel of employees of the organization with various roles such as higher-level managers, project and product managers (of the planned new application), analysts, developers and testers. Each comes with a particular set of needs and questions about the robustness and evolvability of the F/OSS endeavor under assessment. Consequently, the results of the assessment should be presented in such a way that each employee can find answers to the needs and questions associated to his role.

The first three objectives of the task of initiating the assessment reuse the text in the three frames above. It is already inserted as default text in the template found in Annex B. Clearly, this text is generic since it neither mentions a name of F/OSS component nor the scope of the F/OSS endeavor considered by the assessment. Therefore, once a specific F/OSS endeavor is defined and ready to be assess, it is encouraged to substitute or or at least augment this generic text with more specific details (whether imaginary or real). These specific details will make the scenario and the related assessment more understandable and useful. However, it is important that the details do not modify the nature of the F/OSS acquisition which assumes a full F/OSS collaboration.

In the broad context above, it is mentioned that a list of candidate F/OSS components were identified. It was determined that some of their functionality fulfilled the needs of the new software product. During this initial investigation where the available functionality of each F/OSS component was studied, the person performing the inspection likely browsed through the various sites related to each F/OSS component. It is then fair to assume to that during this investigation, the person could have filled the following template with various information on each corresponding F/OSS endeavor. This template is also include as part of Annex B.

Short Name of the F/OSS endeavor (or of the FlOSS component eventually)

- *Official URL:* Central URL of the F/OSS endeavor.
- *Description:* Short description of the F/OSS component planed for integration in the software product. It is also useful to describe the functions of the F/OSS component that will be useful to the new software product.
- *SCM sites:* This lists URL's of source code management sites where many work products produced by the F/OSS endeavor are stored. Furthermore, access control policies and eventually account already created for log in and retrieving work products should also be provided. It is usually useful to give the common command lines (or segment of command lines) to connect and to retrieve information. If several sites exist, it may be useful already to explain why more than one site are available. Are the sites mirrors? or is one site old and another one new? If so, what are their dates of use (begin, end)?
- *Issue/Bug Tracking Systems:* This lists the URL of issue tracking systems. If several sites exist, it may be useful already to explain why more than one site are available. Are the sites mirrors? Is one of the site obsolete? or Is one site old and another one new? If so, what are their dates of use (begin, end)?
- *Mailing lists:* This lists the URL's of mailing lists used by the F/OSS endeavor. It indicates when mailing lists are connected to software processes tools such as version control system or bug tracking system. Such mailing lists are usually read only lists where only tools are allowed to write. Read/Write mailing lists are further labelled based on their concerns or topics: support, announcements, developers, users, etc. New label can be create if necessary.
- *Forums:* This lists the URL's of forums used by community members to discuss various issues related to the F/OSS endeavor. Forums are further labelled based on their concerns or topics: support, users, announcements, developers, etc. New label can be create if necessary.
- *Packaged Distribution List (Releases):* This list the URLs of sites where packaged, downloadable, versionned releases of a F/OSS component are found.
- *Programming Languages:* This lists the various programming language used to implement the F/OSS component

Prior to discuss the fourth objective of initiating the assessment, it is worth mentioning that the template above will be use by Task 3.1 of QualOSS to specify the various FlOSS endeavor to study in WP3.

Finally, the fourth objective of initiating the assessment is concerned with the ***identification of high-level business constraints*** that must be respected by the assessment. Depending on the resources available in the organization, the importance of the software development project, a middleweight assessment on one F/OSS endeavor should take between 1 person-week and 1 person-month.

If the F/OSS component considered is software library, platform or framework and only small contributions are envisaged then the F/OSS assessment needn't take more than one person week. On the other hand, if the F/OSS component to integrate already implements a large part of the functionality then the analysis may take a full 1 person-month, just for the assessment of robustness and evolvability. Toying with the F/OSS component to check its functional behavior is not accounted for in that 1 person-month.

In the particular case of the QualOSS project, WP3 mentions that around 50 F/OSS endeavor will be assessed. In turn, the total effort would vary between 50 person-weeks and 50-person-months, that is between 1 and 4 person-years. Assessments are performed in Tasks 3.2. and 3.3 for which 26 person-months were allocated. Accordingly, this gives around 2 person-weeks to assess a single F/OSS endeavor. This is in-line with the expected time to perform an assessment. We note on the one hand that at the beginning, some quirks in the middleweight assessment method and in the QualOSS platform will most likely be encountered hence increase assessment effort. Furthermore, it will also take time from all QualOSS partners to be fully acquainted with the middleweight assessment method, which also means that at first assessment effort for a single F/OSS endeavor is likely to require more effort. On the other, once the assessment method has stabilized and everyone is familiar with it, assessment are then expected to need less than 1 person-week on the part of the QualOSS consortium. In addition, the validation effort will ask community members to check data and data analysis results and provide feedback. This effort is not include but should only take minimal time to reduce intrusion in the F/OSS communities concerned.

*5.1.2.2.*Setting Up and Planing an Assessment for a Full FLOSS Collaboration

Referring to Section 4.2.2, we find that the objectives of the setting up and planning task are 1. to identify accurately the scope of the F/OSS endeavor under assessment, 2. to select the evaluators and identify other people who will be consulted during or might be impacted by the assessment, 3. to identify workflow to follow during the remaining of the assessment activity and the other rules to obey when performing operations of the workflow, 4. to select the appropriate procedures and tools to use to perform each operation of the workflow, 5. to plan the supervision strategy and inform the person in charge of supervising the assessment of this strategy, and 6. to validate the outcomes produce by the first 5 objectives.

To help an assessment to achieve these objectives, in the following subsections, the standard QualOSS assessment method first presents the workflow to follow during an assessment. Every operation of the workflow is described. Second, the scoping of an F/OSS endeavor is explained. Furthermore, it also recalls what must be done to satisfy the traceability requirement when scoping an FlOSS endeavor. Third, a few remarks about the type of expertises required from people using the standard QualOSS assessment method. These first three points respectively show how the standard QualOSS assessment method assist in fulfilling objectives 1, 2, and 3 of this setting up and planning task. Concerning objective 4, we highlight that the generic workflow, illustrated in Figure 5, includes operations 9 and 10 where procedures and tools for the assessment are selected and then configured. Concerning objectives 5 and 6, rules to obey during an assessment that are specified below in Section 5.1.2.2.2, added to the description of operations of the workflow that implements these rules together, enforce that supervision and validation take place in a methodical way and that the traceability requirement is guaranteed.

5.1.2.2.1 Workflow of an Assessment

Figure 5 presents the workflow of operations to follow during an assessment. This workflow is imposed by the standard QualOSS assessment method.

The entry point of the workflow assumes that the assessment report template used as input of operation 1 already contains the results from the previous task of initiating an assessment. As part of the Setting up and Planning Task, the operations in the bold box are to be executed.

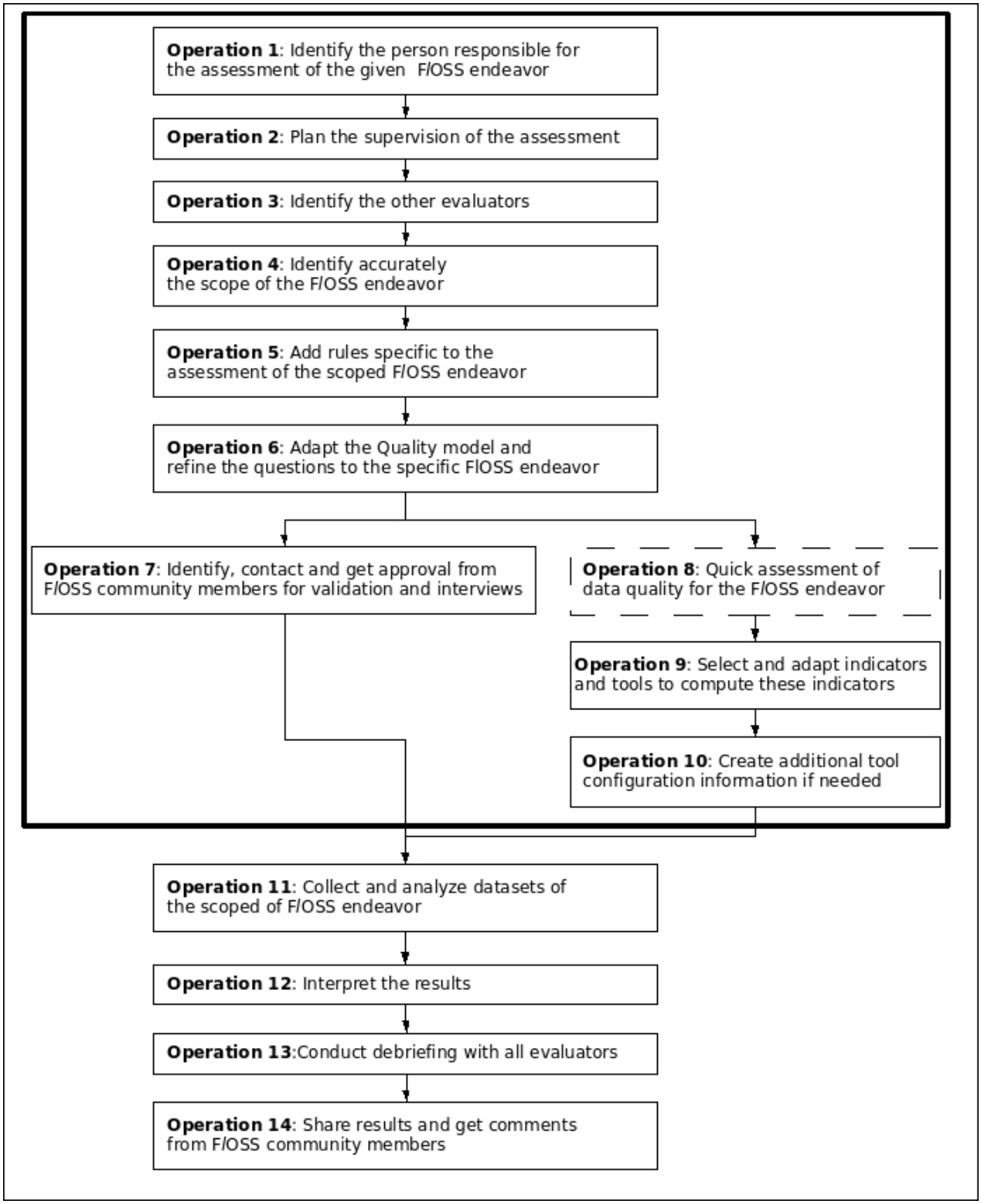

Figure 5: Workflow of operations imposed by the middleweight assessment method

Below we explain each operation of the workflow. In particular, the inputs, outputs and processing are described.

Operation 1: Identification of the person responsible for the assessment of the given FlOSS endeavor.

Implements Tasks:	Setting up and Planning an Assessment
Inputs:	
• Fact sheet about the F/OSS endeavor to assess • Template of the assessment report with results from the task initiating an assessment	
Outputs:	
• Report instantiated from the template with the responsible person name specified and other eventual information know at the beginning of the assessment	
Brief Description of the operations:	
The responsible person uses the template assessment report to create an initial (incomplete) assessment report for the given F/OSS endeavor. The responsible person already edit the generic report with the know information such as the short name of the FlOSS endeavor, the modified introduction with additional input concerning the acquisition of the given F/OSS endeavor. Furthermore, the data from the existing fact sheet is inserted in section 1.2 of the assessment report, and the responsible person also insert her or his name on the title page of the assessment report.	

Operation 2: Plan the supervision of the assessment

Implements Tasks:	Setting up and Planning an Assessment
Inputs:	
• None	
Outputs:	
• Prepare the empty supervision report and add the rules to obey during the assessment	
Brief Description of the operations:	
In this middleweight assessment, the supervision is handled by the person responsible for the whole assessment. This person will simply review the rules of the assessment to make sure that he understand clearly the responsibility regarding the supervision of the assessment. Second, he prepares the supervision report. This report should contains a first section with the rules that all evaluators will respect during the assessment. Furthermore, it also includes two empty sections: first, to collect data regarding unanticipated or low level actions that had to take place but were not anticipated in the initial workflow and second, to transcribe the debriefing information at the end of the workflow. We note that this supervision report is a second report that may be added as an annex of the main assessment report if desired. The report is then sent to the person who supervised the assessment however in this middleweight assessment method, the supervision is also handled by the person responsible for the whole assessment.	

Operation 3: Identify the other evaluators

Implements Tasks:	Setting up and Planning an Assessment

Inputs:
• Assessment Report (from operation 1) • Supervision report with rules (from operation 2) • Table Template of Annex C for recording low-level or unexpected
Outputs:
• Workflow instantiated to this assessment where all operations have been assigned to specific evaluators
Brief Description of the operations:
Evaluators are selected based on the expertise needed for the assessment, for instance, knowledge of the Python Language in order to launch the test of the given F/OSS components. At the end of this operation, each workflow operation has been assigned to one or several specific evaluators who have agreed to conduct the given operation. Eventually, the people responsible for validating the outcome of each operation of the workflow must be identified. Furthermore, the person responsible for the assessment must remind all evaluators the rules of the assessment given in the supervision report.

Operation 4: Identify accurately the scope of the FlOSS endeavor

Implements Tasks:	• Setting up and Planning an Assessment • Supervising an Assessment
Inputs:	
• Fact sheet of the FlOSS endeavor in the assessment report • Supervision report (from operations 3)	
Outputs:	
• Accurate list of work products related to the product (source code, build, test, documentation) • List of data sources and work products available for collecting information related to Community members • List of data sources and work products available for collecting information related to Software Processes • List of data sources and work products available for collecting information related to Tools • Supervision report with eventual additional actions recorded	
Brief Description of the operations:	
The scope of work product related to the product must be clearly identified since the product, test and documentation are crucial in the full F/OSS collaboration scenario. Although it is important to be as accurate as possible when scoping the datasets (URL) related to community members, software processes and tools, it is also possible to keep a wider scope than truly needed and it will be the duty of the person who actually analyzing need to assess characteristics of the community, processes or tools to record the data accessed in order to analyze and evaluate each of the characteristics. In parallel to determining the scope, it is as important to record how it was done. For instance, if a script is written to extract the name of all the test files from a version control or a packaged distribution then it is important to keep a trace of that script. This is done	

by recording the script name and assigning it a URL using the action recording template given in Annex C.

Operation 5: Add rules specific to the assessment of this FlOSS endeavor

Implements Tasks:	• Setting up and Planning an Assessment • Supervising an Assessment
Inputs:	
• Supervision report with list of rules included	
Outputs:	
• Supervision report with list of rules specific to the given F/OSS endeavor	
Brief Description of the operations:	
Based on the scope of the particular F/OSS endeavor, based on the number of evaluators it may be possible to adapt the rules. The normal situation is never to erase rules but only to add more rules that are needed due to the specificities of the given F/OSS endeavor. For instance, the bug tracking system on SourceForge for the given F/OSS endeavor will be ignored because it is not really used by the F/OSS endeavor. It is important that all evaluators review the final set of rules and agree with them.	

Operation 6: Adapt the quality model and refine the questions to this specific FlOSS endeavor

Implements Tasks:	• Setting up and Planning an Assessment • Supervising an Assessment
Inputs:	
• Assessment report with the scoped F/OSS endeavor • Supervision report with list of rules	
Outputs:	
• Assessment report with the specific quality model to use for the summarizing the results of the assessment of the F/OSS endeavor. • Supervision report with eventual additional actions recorded	
Brief description of the operations:	
In the context of QualOSS, the quality model used is the one given in Annex A. It will not will not be adapted when assessing the F/OSS endeavors in WP3. The QualOSS platform will come with the quality model presented in Annex A. However, we add that it is possible to customize that quality model, for instance, with new characteristics or more specific questions, if desired. This dimension will be explored during the case studies in WP5. In case adaptation must implemented in the context of WP3, it is crucial to record those adjustment using the action recording template given in Annex C. Furthermore a concrete reasoning must be given as to why the standard QualOSS quality model and questions had to be modified for the assessment.	

Operation 7: Identify, contact, and get approval from FlOSS community members for validation and interviews

Implements Tasks	• Setting up and Planning an Assessment • Supervising an Assessment
Inputs:	
• Assessment report (from operation 6) • Supervision report (from operation 6)	
Outputs:	
• Assessment report where list of FlOSS community member who will help with data validation and other interview needs • Supervision report with eventual additional actions recorded	
Brief Description of the operations:	
Based on the accurate scope of the F/OSS endeavor, it will then be possible to identify a few relevant F/OSS community members who could help us with data validation as well as comment on assessment results in the last operation of the workflow. This validation correspond to the second round validation presented in Section 6.1. However, not all F/OSS community members will have the time or interest to look over assessment results. In turn, during this step, they will be contacted and they will be explained what will be expected from them. If they agree they will then be added to the list of participant to the assessment. It will be hard to raise the interest to many F/OSS community members in turn it is important to record the various actions taken in order to have concrete traces of the effort undertaken to find F/OSS community members who would be willing to perform validation work. The actions must be recorded using the action recording template given in Annex C	

Operation 8: [OPTIONAL] Quick assessment of the data quality for the given FlOSS endeavor

Implements Tasks	Setting up and Planning an Assessment
Inputs:	
• Assessment report with accurate F/OSS endeavor scope (from operation 6)	
Outputs:	
• Evaluation of data quality for Work products • Evaluation of data quality for Community Members • Evaluation of data quality for Software Processes • Evaluation of data quality for Tools and Dependencies	
Brief Description of the operations:	
This operation consists of analyzing the various data source and their specific URL to verify if the data quality needed to perform our assessment is present. This quick evaluation of data quality will remain very light. It should take between 2 and 4 person-hours at most. Data quality will impact assessment result. In particular, how trustworthy these results are? and how much effort it will take to analyze the datasets to evaluate the various characteristics. For example, if issues are reported in a mailing list rather than a	

bug tracker, it will take more effort to analyze them and evaluate characteristic that need measures on bugs. However, a mailing list is better than nothing. This evaluation on data quality will be include in the assessment report and brief explanation of the impact of the data quality should also be presented in the assessment report. For instance, explaining how the given data quality impacts the assessment results.

Operation 9: Select and adapt indicators and tools for computing these indicators

Implements Tasks	• Setting up and Planning an Assessment • Supervising an Assessment
Inputs:	
• Assessment report with scoped F/OSS endeavor and data quality evaluation • Supervision report (from operation 6)	
Outputs:	
• Specific selection of the indicators and of tools for computing these indicators • Supervision report with eventual additional actions recorded	
Brief Description of the operations:	
Based on the data available, their format and their content quality, it is possible to determine what indicators can and cannot be computed automatically. In the case where an indicator cannot be computed automatically, it may be possible to propose an alternate manual procedures. Although less reliable, it will at least propose a result. During this operation, evaluator also select the indicator to use and the tools to compute them. Should the reader wonder how the indicator computed its results, it is important to include the definition of the indicators used in an assessment as part of the assessment report. However, such more formal definition should only be insert in an annex of the assessment report. The template in Annex D of this document, which was already used during Task 1.5, may be filled to present the definition of each indicator. We note that ready-made indicators will be distributed with the QualOSS platform. However, these indicators will only result from Task 4.2 hence they are not yet presented in this deliverable D4.1. It is currently not anticipated that the assessments performed during Work Package 3 will alter indicator. However, if indicator tailoring is needed then it is crucial to record it using the action recording template given in Annex C. In addition to the tailoring, a concrete reasoning must be provided when filling the template to explain why the adjustments to the standard QualOSS indicators had to be performed for the given assessment.	

Operation 10: Create specific tool configuration information (if needed)

Implements Tasks	• Setting up and Planning an Assessment • Supervising an Assessment
Inputs:	
• Assessment report with indicators (from operation 9) • Supervision report (from operation 9)	

Outputs:
• Configuration information needed by tools and procedures of indicator computation • Supervision report with eventual additional actions recorded
Brief Description of the operations:
In certain cases, analysis tools or manual procedures must be configured properly in order to return reliable, accurate results. For example, in order to to perform its analysis correctly, an analysis tool may need to know the libraries used by a F/OSS component. This is usually the case when analysis tools perform their analysis on object code rather than source code. This means that given the scope FlOSS endeavor and knowing the tools needed to compute indicators, it is now possible to create the necessary configuration of tools. Rather the attach the raw configuration information to the assessment report, it may be more important to package the configuration and store it in a particular, retrievable URI and only mention that URI in the assessment report. All tools and procedure configuration must be persisted and assigned URI. The URI must then be recorder using the action recording template given in Annex C.

Operation 11: Collect and Analyze scoped datasets of the FlOSS endeavor

Implements Tasks	• Collecting and Analyzing Data • Supervising an Assessment
Inputs:	
• Assessment report with quality model and indicators (from operation 10) • Data Validation (from operation 7) • Supervision report (from operations 7 and 10)	
Outputs:	
• Analyzed data stored in a database	
Brief Description of the operations:	
In the context of QualOSS, this step consists of launching the QualOSS platform so it perform all necessary automated measurements. Furthermore, the assigned evaluators will follow the appropriate manual procedures to assess each sub-goal (or characteristic) that do not have automated means to be assessed. Naturally, they must record all their assumptions and decisions. Once they obtain an assessment score, they will be able to insert it in the QualOSS database via a special feature of the QualOSS platform. The supervision report from operations 7 and 10 must be concatenated. Furthermore, it is crucial to record any problem with the QualOSS platform, and how it had to be modified to execute the data collection and analysis properly. As mentioned above, for manual procedure, evaluators must record all their assumptions and decisions using the action recording template given in Annex C. Depending on how broad the scope of datasets identified in operation 4, it will often be necessary to record the scope of the dataset collected and analyzed. For example, tedious manual procedures will often assign scores based on a small data sample, it is important that all samples used for assessing various sub-goals of the quality model be recorded.	

Operation 12: Interpret the results

Implements Tasks	• Interpreting the results • Supervising an Assessment
Inputs:	
• Assessment report with quality model and indicators (from operation 10) • Measure Database (from operation 11) • Supervision report (from operation 11)	
Outputs:	
• Assessment report with the complete assessment results • Supervision report with eventual additional actions recorded	
Brief Description of the operations:	
Combining the quality model already specified in the assessment report and the measurement results, it is possible to obtain answer to questions and using this answer it is also possible compute indicators values. These indicator values will propose one or more ways to look at a particular sub-goal (or characteristic) of the quality model, in turn each user will need to decide which indicators are more relevant or have priority in his case. This step will be aided by the interpretation manual, which will be a result of Task 4.2 and presented in deliverable D4.2. In the context of QualOSS, indicators values will be used to indicate the risk related to a sub-goal (or characteristic) in the particular context of the full F/OSS collaboration scenario. Based in this scenario, indicators will by default be assigned different priorities which may then influence the risk level computed. However, if one disagrees with the default priorities changing them in the QualOSS platform will not be complicated. If priorities are changed for the assessment of a particular FlOSS endeavor, it must be recorded using the action recording template in Annex C. Furthermore, a appropriate reasoning must also be given as to why the change in priorities had to be made.	

Operation 13: Conduct debriefing with all evaluators

Implements Tasks	• Interpreting the results • Supervising an Assessment
Inputs:	
• Assessment report (from operation 12) • Supervision report (from operation 12)	
Outputs:	
• Final supervision report with debriefing information and recommendations for future assessments • Final Assessment Report of the F/OSS endeavor.	
Brief Description of the operations:	
Based on the action previously recorded, the person responsible for the assessment will propose an argument supporting the validity of the assessment. On the other hand, if many unanticipated, corrective actions had to be taken, but recording actions are missing then the validity of the assessment is jeopardized and the responsible person	

should then indicated that information of the assessment report where appropriate. In the context of QualOSS and in particular WP3, the responsible of the assessment should concatenated the supervision report to the final assessment report

Operation 14: Share the results and get comments from the FlOSS community members

Implements Tasks:	• Interpreting the results • Supervising an Assessment
Inputs:	
• Final assessment report (from operation 13)	
Outputs:	
• Relevant comments from F/OSS community members appended to the assessment report	
Brief Description of the operations:	
F/OSS community will be given the opportunities to comment on the results of the assessment. In some cases, they will agree with the risk level assigned to an given sub-goal of the quality model while in other cases, they may find a justification for their weakness. Yet, they may also show a complete disagreement regarding the useful of certain indicators. As long as explained and presented in a respectful and argumentative way, these comments will be included in the assessment report. It may also be possible that in the light of the assessment results, the F/OSS community member consulted mention that the datasets used were incomplete. If additional data sources are available to complete datasets, then this is ground for dismissal of the assessment results. However, it should be determined why these data sources were not mentioned during operation 7.	

5.1.2.2.2 Rules of the Standard QualOSS Assessment Method

Below are the rules to respect when performing the assessment of an F/OSS endeavor based on this middleweight assessment method. Although this middleweight assessment method is design to be used during the QualOSS project, it could also be outside that context, for instance, used directly by an enterprise that desires to follow the QualOSS methodology to assist it during software acquisition in a full F/OSS collaboration scenario. Hence, we distinguish between two types of rules. In particular, the rules whose identifier only contains a number are mandatory rules of this middleweight assessment method. Rule whose identifier contains a Q only apply to assessment performed during the QualOSS project.

Rule 1. Every assessment performed using this middleweight assessment method must have a responsible person assigned.

Rule 2. The responsible person is in charge to verify that the workflow proposed for the assessment of an F/OSS endeavor is respected.

Rule 3. The responsible person is in charge to verify that each task reaches its objectives,

Rule 4. The responsible person is in charge to verify that the traceability requirement is satisfied throughout the assessment activity.

Rule 5. The responsible person is in charge to verify that the outcomes of each task are validated before handing them to the other tasks of the assessment.

Rule 6. Every assessment must use the quality model presented in Annex A including the questions attached to each leave of the tree hierarchy. Ignoring part of the quality model hierarchy in an assessment must be justified.

Rule 7. The person responsible for the assessment must conduct a debriefing with all evaluators; the result of this debriefing must collect all information regarding unanticipated actions that needed to take place during the assessment. Furthermore, it also records the tension points that happened during the assessment.

Rule Q1. Results of test runs performed on datasets during this setting up and planning task may not be used for the final assessment. They will need to be regenerated properly following the workflow sequence of operations specified by the middleweight assessment method.

Rule Q2. All assessment results will be stored on the QualOSS platform hosted at CETIC and where all QualOSS consortium active participants have an account.

Rule Q3. When possible, assessment results will be generated using the QualOSS platform hosted at CETIC.

Rule Q4. Every assessment must use the indicators with the threshold provided the quality model presented in Annex A including the questions attached to each leave of the tree hierarchy.

Rule Q5. The outcome of operations of the workflow must be verified and validated by by at least three people belonging to at least two different organizations of the QualOSS consortium.

Rule Q6. Assessment information collected during an assessment are all gathered in a single document called the assessment report.

Rule Q7. The modified assessment report resulting from operations of the workflow is shared with others through the Trac website: http://qualoss-partners.libresoft.es, in particular using the wiki in connection with Trac tickets. In particular, all operations will be assigned to the person responsible of the assessment. Other evaluator will be in CC of the ticket. The responsible person will then distribute the work and eventually reassign tickets if needed.

5.1.2.2.3 Scoping an F/OSS endeavor

First, we recall that scoping an F/OSS endeavor consists of specifying the work products, the community members, the software processes, and the tools and dependencies to study during an assessment. Second, we suggest analyze the business usage scenario for which a assessment method is built. It naturally induces how to initiate the scoping of a F/OSS endeavor.

This middleweight assessment method is built for the full F/OSS collaboration scenario described in Section 3.2.1.2 where the main purpose is to decide if an F/OSS component should be integrated in an software product. The FlOSS component is therefore the starting point of the scoping operation. For instance, the scoping will start from the source code without neglecting other work product such as binaries, documentation, and test files related to that source code. For example, to specify the scope related to the *Eclipse BIRT reporting engine*, one may list a subset of the source code files found in a packaged

distribution for the entire Eclipse BIRT plug-in, then for documentation work products, the relevant API in HTML files and tutorial files will be selected. Finally, the unit test found next to the source files may then be listed for the tests.

Second, the dependencies of the F/OSS components will likely be of interest in turn, a certain subset of work products from those dependencies will also be added to the scope so they will also be inspected during the assessment. We note that for these dependencies the work products to include in the scope is usually a reduce set of information however, they will often include other data that merely the code. The quality model presented in Annex A shows the characteristics of dependencies that need to be assessed when following this middleweight assessment method. Based on the definition of the characteristics, F/OSS endeavor compatibility, F/OSS endeavor lobbyability and FlOSS endeavor referenceability, it is clear that data other than source code will need to be included in the scope to analyze during this assessment.

Third, restrictions on the software processes and the community members could be considered in the scope. However, in the full F/OSS collaboration scenario, no restriction will put on community members or software processes. This means that all information needed to evaluate the characteristics of software processes and of community members listed in Annex A must be added to the scope.

The first step of scoping can often mostly be automated. That is, the packaged distributions or the directories in a version control system provided by the F/OSS endeavor directly provide the needed scope or at worst a script can be written to selected only a subset of files for that scope. Problems may however occur if the F/OSS endeavor lack the data needed to define the desired scope, for instance, if older versions of the code or of documentation do not exist and the scope desired would have included that data.

The second and third step of scoping a F/OSS endeavor are usually procedures more manual that can however be assisted by some automated tools. Furthermore, databases from other F/OSS research should also be exploited to assist with these two steps. Most notably, FLOSSMETRICS and FLOSSMole databases may contain data about F/OSS community members and other information, which may transitively be used to identify data need to assess software processes. However, due to the lack of conventions on where to store information related to information on dependencies, community members and software processes coupled to the unstructured nature of information dependencies, community members and software processes, the scope on all three will involve a significant manual effort that could amount to several person-hours of work.

Alternatively, one may initially provide a scope larger than needed for datasets of dependencies, community members and software processes and then only identify on-demand the elements required from these datasets in order to assessment dependencies, community members and software processes. For example, the initial scope for software process may be defined as any URL transitively accessible from the information given in the fact sheet of an F/OSS endeavor (such as those presented in D3.1). Then, it will be the task of the person who assess the maturity of the various software processes to clearly record only the URL considered for assigning maturity levels to each software processes.

This alternative way of scoping where a super-scope is initially defined and then restricted later on is more pragmatic but it requires a constant vigilance on keeping traces of URL visited when an evaluators performs a manual procedures for evaluating a particular characteristic of the quality model presented in Annex A.

5.1.2.2.4 Expertise Required

In brief, this middleweight assessment method requires a high degree of rigor.

First, it is important for the person responsible of an assessment to be very methodological. Furthermore, if the assessment is performed by a team of people, which is often the case in real-life situations, then the responsible person must also have people management skills to guarantee that the rules, specified below, are respected through out an assessment activity. A responsible person should usually have an experience with managing short, effort-intensive projects.

The QualOSS platform simplifies the tedious tasks of an assessment and also enforces a high degree of rigor. However, it does require technical expertise to be installed, configured, and launched. In turn, the standard QualOSS platform user is expected to have the profile of a software developer. Incidentally, this platform is not targeting lambda users who expect to point and click for installing, configuring and executing assessments. Due to the non-standard and unstructured nature of F/OSS endeavors datasets added to the huge variation in data quality between F/OSS endeavor, an assessment cannot be oversimplified. Oversimplification would run the risk to provide bad assessment results or only very limit valid assessment information with little added value.

5.1.2.2.5 Quality Model, Indicators, Tools and Procedures

Beside scoping an F/OSS endeavor, other important and tedious efforts of the setting up and planing task are operations 6, 9 and 10, which respectively ask to identify the quality model for the assessment, to select the indicators to assess leaf goals of the quality model as well as the tools and procedures for computing indicator values, and then configure these tools and procedures properly so the QualOSS platform will work with them.

The standard QualOSS assessment method imposes the quality model to use for the assessment of the Work Package 3. The quality model is presented in Annex A. We highlight that the quality model follows the Goal-Question-Metrics paradigm and therefore, the definitions of quality characteristics are further expressed as assessment goals in direct relation to the full F/OSS collaboration scenario. Moreover, questions to answer in order to evaluate the degree to which each assessment goal is satisfied are listed in Annex A.

Similarly, the standard QualOSS assessment method will also proposed a set of indicators to use for assessing each leaf goal of the quality model. Indicators for each goal of the quality model are currently being elaborated. They will be presented in deliverable D4.2.

Based on a brief evaluation of the data quality (performed in the optional operation 8), it will be possible to determine before the full collection and analysis of data whether all, some or none of the measurement will be able to take place. If only some or none of the measurement will be taken, for instance, due to lack of data or data of poor quality, it may then be possible to specify how to handle and adjust the assessment to account for that situation. For example, if the data is unstructured, it may be possible to perform a manual measurement on a small sample of the dataset and still obtain a measure at the end. However, it is clear that the resulting measure should be annotated as being imprecise.

It is also worth noting that measures are used by indicators and indicators assess risk of an F/OSS endeavor in particular context of the full F/OSS collaboration acquisition scenario. In that situation, lack of data or data of poor quality will often be judge as highly risky. In turn, it is also acceptable not to take the measurement, which will in turn translate as risk indicator values.

5.1.2.3. Collecting and Analyzing Data for an Assessment in a Full FLOSS Collaboration

The task of collecting and analyzing data correspond to the operation 11 of the workflow. From the assessment process description, this task has the 4 main objectives:

First, to collect the data whose sources were identified when setting up the assessment, in particular, the accurate scope dataset related to the F/OSS endeavor.

Second, to analyze the data based on information provided in the quality model in Annex A and the indicators that are being developed during Task 4.2.

Third, to maintain a link between the raw data, the methods and tools used to collect and analyze that data, the results of data analysis. For the analysis tools connected to the QualOSS platform, the traceability will automatically maintained. However, for tools not yet connected or for manual assessment procedures, the

traceability between input, output and processing is not automated but it is enforced. Indeed, It is the job of the evaluator who ran the tool or executed the procedure to enter that information in the QualOSS database of assessment to which the QualOSS platform is connected but to protect its integrity, the QualOSS database requires that information about the input, output and processing be entered before accepting an evaluation score or measures.

The fourth objective is to validate the collected and analyzed data. In the standard QualOSS assessment, this validation is twofold. First, rule Q5 requires that all measures be validated by two other person than the one who performed a measurement. Second, the measures will be validated with F/OSS community members part of the F/OSS endeavor under assessment. This validation corresponds to the third round validation mentioned in Section 6.1.

Furthermore, this task also has additional objectives when problems are encountered during the collection or the analysis of data. In particular, all problems that happen when collecting or analyzing data must be recorded following the template in Annex C. It is also the responsibility of this task to adapt and validate the procedures and tools that caused the problem. Alternatively, it is also possible to explain that no tools or procedures could be design to collect or analyze the anticipated datasets. In such cases, it is also important to record using the template in Annex C what was attempted, what was not attempted and the reason why these attempts failed. It is also possible to simply indicated that the dataset was much poorer than anticipated from Operation 8.

Achieving all the objectives of this task is crucial for the validity of the whole assessment. As a consequence, the person responsible for the assessment must explicitly subdivide the effort using Trac tickets in order to guarantee traceability. Naturally, these tickets must only be create if there is a division of labor for collecting and analyzing data among several evaluators. Otherwise the responsibility is on the chief evaluator responsible for the whole assessment. In any case, as requested in rule Q5, the outcomes of this task must also be validated by two other evaluators.

5.1.2.4. Interpreting the Results of an Assessment in a Full FLOSS Collaboration

The task of interpreting the result of an assessment correspond to the workflow operations 12 and 14. From the assessment process description, this task has three main objectives:

First, to interpret the analyzed data resulting from the task *collect and analyze data* (or operation 11) using the interpretation methods selected during the task of *setting up and planning an assessment*, in particular using the outcome of operations 6 and 9, which respectively specified the quality model and indicator to use for interpreting the results.

The second objective consists in maintaining a link between the analyzed data, the interpretation methods, and the resulting interpretation. In practice, this will be achieved by storing the quality model and indicators in the QualOSS database when launching an assessment of an F/OSS endeavor with the QualOSS platform.

The third objective, which correspond to operation 14 of the workflow, consists in validating the interpretation of made on the analyzed data. In practice, the interpretation is performed by applying indicators on collected and analyzed data. This validation is twofold. First, rule Q5 requires that each interpretation be validated by two other people than the one who initially computed the indicator value. Second, the interpretation will be presented to F/OSS community members part of the F/OSS endeavor under assessment. This will be useful to gather the feedback. However, to keep this exercise useful, it is important that F/OSS community member understand the business context for which assessment results are presented. In other words, they must clearly understand that indicators are not indicating theoretical good or bad but rather just an instrument used by enterprises to determine what are the potential risks with an F/OSS endeavor whose F/OSS component is considered for integration a software product while maintaining a full F/OSS collaboration.

This task also has additional objectives when problems are encountered during the interpretation of results. In particular, all problems that happen when interpreting results must be recorded following the template in Annex C.

When problems are encountered, this task may decide to solve them. In particular, errors in interpreting results may be due to lack of data for compute indicators or unanticipated data analysis results for which current indicators are not appropriate. When proposing new indicators, it is the responsibility of this task (or rather, of the chief evaluator of an assessment) to make sure that these new indicators are validated following the same process as for the other indicators. The process for validating an indicator in the context of QualOSS is presented in Section 6.3.2.

*5.1.2.5.*Supervising an Assessment for a Full FLOSS Collaboration

The task of supervising an assessment runs during the whole assessment. This objectives are:

First to monitor that the tasks are executed as planned including that each task performs the validation of its outcome. Instead of having additional workflow operations to reach this objective, the standard QualOSS assessment method distributed the supervision actions in every workflow operations and also emphasized it through explicit rules. For example, rule Q5 requires for the outcome of every workflow operation to be validated by three different people. Furthermore, to make sure that the traceability requirement of the assessment is fulfilled, most workflow operations are ordered to record complex or unanticipated actions by filling the template in Annex C. Rule Q7 requires that every workflow operation be registered as a ticket in order to guaranteed a traceable supervision of the whole workflow.

Given the time consuming operation 11 of collecting and analyzing data, it is likely to be distributed among several evaluators. The supervision principle again requires that the division of labor be explicitly specified by creating and assigning ticket to these various evaluators who will also be responsible to record complex or unanticipated actions using the template in Annex C.

The second objective is of an overarching nature. It asks to identify tensions between the activity elements that occurred during an assessment. In particular, as mentioned in Section 4, the activity elements are:

- (subjects) The evaluators,
- (object) The data of the F/OSS endeavor,
- (tools) The procedures and tools including the quality model, indicators used in the standard QualOSS assessment method and even the Trac ticketing and wiki system,
- (community) the other evaluators and F/OSS community members who are involved in the assessment,
- (rules) the rules imposed by the standard QualOSS assessment method
- (division of labor) the workflow imposed by the standard QualOSS assessment method

The standard QualOSS assessment method will be used significant amount of time to evaluate different F/OSS endeavors. Consequently, it is our goal to improve it over the course of each assessment (or every 5 assessments). In order to improve it in a constructive way, it is important to identify the tensions that arose between the elements mentioned above. It is therefore our intention to conduct interviews with the evaluators at the end of assessments. This is even more critical when an assessment encountered many unanticipated problems.

In the standard QualOSS assessment method, the plan is for the chief evaluator to conduct informal interview with all the evaluators at once during a conference call. During that conference call, every elements

and its interactions with the other elements will be discussed and summarized in a Trac ticket, in particular the Trac ticket create for operation 13 of the workflow. If during that interview, it is determined that big problems occurred during the assessment of one or several F/OSS endeavors, and that such problems jeopardize the validity of the assessment results then it should be directly reported the QualOSS consortium. First, this will avoid presenting these results to F/OSS community members who would waste their time commenting invalid results in operation 14. More importantly, the QualOSS consortium will debate on how to adapt the standard QualOSS assessment method in order to eliminate these big problems.

5.2 Lightweight Applications of the Standard QualOSS Assessment Method

Creating a particular QualOSS assessment method from scratch such as for creating the standard QualOSS assessment method presented in Section 5.1 takes considerable time. Even simply following the standard QualOSS assessment method rigorously can often take an effort of more than 1 person-week to assess a single F/OSS endeavor.

Clearly, one could propose a less rigorous assessment method where limited traceability and validation would be performed. However, that method would most likely not meet the requirements set by the QualOSS methodology in order to be considered a QualOSS assessment method. Therefore, this method could not claim to be a *QualOSS* assessment method.

Credible engineering disciplines expect the utter most rigor. Furthermore, traceability is also expected in these disciplines. Software Engineering should therefore follow the example of its older siblings. In other words, to gain in recognition, Software Engineering should clearly distinguish between quick cook books versus its recognized method. We believe that the requirements set by the QualOSS methodology are in the right direction, and therefore loosening them would merely provide yet other quick receipt with very little value addition for Software Engineering and Computer Science in general. Furthermore, other less rigorous assessment methods already exist such as OpenBRR and QSOS, and untraceable information on F/OSS projects may also be found on sites such as Ohloh. We therefore direct people who do not need a real rigorous assessment method to these other works.

In the lightweight application, the goal is to obtain assessment results in around one hour to half a day. The main point of this lightweight application is to leverage on existing assessment results. However, this is not the only point, the lightweight application of the standard QualOSS assessment method is also of its own right, a lightweight QualOSS assessment method. In turn, it must satisfy all the requirements set by the QualOSS methodology.

The lightweight QualOSS assessment method could briefly be described as a method to educate an individual to read, eventually slightly adapt and then interpret existing assessment results obtained by others who have rigorously followed the standard QualOSS assessment method. Indeed, if a training step is not taken then chances will be too great that a reader of the assessment results would only focus on the brief summary of the assessment results and therefore forget to verify the F/OSS acquisition scenario for which these results are meant. Consequently, if the reader's F/OSS acquisition context is different then the one used to obtain the assessment results then the reader may be ill advised by these results.

As expressed in the situation described above, the context of a lightweight QualOSS assessment method assumes that only a single individual will be involved. Clearly it will be that person's responsibility to meet all the requirement set by the QualOSS methodology in order to create his own lightweight QualOSS assessment method. Creating a new lightweight QualOSS assessment method requires describing it in a document. That document will have to briefly mention how this new lightweight QualOSS assessment method satisfy the requirements set by the QualOSS methodology.

Requirements 2-5 are easy to satisfy, for instance, mentioning the version of the QualOSS methodology used, in this case, will simply state the one used by the standard QualOSS assessment method.

Furthermore, since a lightweight QualOSS assessment method relies on assessment results obtained by another QualOSS assessment method, it must mention the name and version of the QualOSS assessment method on which it relies.

Requirement 6 mentions that all QualOSS assessment methods must implement the assessment process prescribed by the QualOSS methodology. The subsection below explains how lightweight QualOSS assessment methods are expected to implement that assessment process.

Requirement 7 regarding traceability is much easier to satisfy since it is already satisfied by the existing assessment results used by the QualOSS lightweight assessment method. Furthermore, traceability is much simpler when only a single individual is involved in a lightweight assessment.

5.2.1 Implementing the Assessment Process for a lightweight QualOSS Assessment Method

This section briefly presents how a lightweight QualOSS assessment method is expected to implement the assessment process prescribed by the QualOSS methodology.

It is expected that following a lightweight QualOSS assessment method to obtain assessment results should require around less than one hour. Naturally, the lightweight QualOSS assessment needs to be documented prior to being followed. It is expected that describing a lightweight QualOSS assessment method should only take around 2 to 3 hours.

The other objectives of initiating an assessment require the description of the broad context, in which the assessment takes place, to justify the need for the assessment and also present how the results of this assessment will be used. Describing the broad context and justification for the assessment should take between 20 and 30 minutes.

Based on this description, the evaluator needs to determine if the assessment context he desires and the assessment context of the existing assessment results are similar enough in order to use these existing assessment results in this new, desired assessment context. For instance, the assessment results obtained using the standard QualOSS assessment method assume the full F/OSS collaboration context.

If the context in which one wants to use results from that standard method are quite different from the full F/OSS collaboration acquisition scenario then the results will not apply and could even prove inadequate to the new desired context. In such a case, the evaluator will require a heavyweight approach which can be helped by following the guidelines presented in the next section. However, the evaluator should realize that a much larger amount of effort will likely be required to obtain the new QualOSS assessment method.

On the other hand, if the broad context and the need for the assessment are similar to the ones used to obtain the existing results then the evaluator will be able verify that the existing results he plans to use address the same F/OSS endeavor scope that the one desired by the evaluator. For instance, if existing results are only for the Eclipse BIRT engine then the evaluator should not assume the same results would be obtained for the whole Eclipse BIRT plugin from which the engine is just a part. Only if the scope desired is the same as the one treated in the assessment results can these results be reused in a lightweight QualOSS assessment.

The setting up and planing phase are much simpler than for the standard QualOSS assessment method but it will require an intensive effort. In particular, the evaluator will be asked to review the quality characteristics assessed in the existing results and verify whether they are all relevant. If not, the irrelevant characteristics may be eliminated. We note that when creating a lightweight QualOSS methodology, not characteristics may be added, characteristic may only be removed from the list. It is however possible to prioritize the characteristics.

The second step of the setting up stage consists of studying the questions used to obtain the existing results and how indicators aggregates answers to these questions to evaluate the quality characteristic. If questions are irrelevant, they can be eliminated however as for the characteristics no new questions may be added. With regards to indicators, the evaluator is free to adapt them to his context, that is, to specify how he believes risks scores should be computed in his context. Once, the quality characteristic, the questions and indicators have been adapted to the evaluator's context and described in the document presenting the new lightweight QualOSS assessment method then the setting up and planing phase is done.

The collection and analysis of data relies on the analyzed data from the existing assessment results. Hence this task does not require any effort from the evaluator conducting a lightweight QualOSS assessments. Indeed, this is the reason reason why a lightweight QualOSS assessment can be done between 1 hour and half day. Furthermore, since the analyzed data and measures are taken from the existing assessment results, there is no surprises such being of poor quality (incomplete, noisy, unavailable, ...)

The task of interpreting the results simply follows the decision taken during the setting up and planing task regarding the quality characteristic to asses, the questions and the indicators selected to assess each of these characteristics. Since the analyzed data on which to apply the indicators are known, there are no suprises, such as whether a indicator can be computed or not. In the case of a lightweight QualOSS assessment indicators can always be computed.

The last task is that of supervision of the assessment. In a lightweight QualOSS assessment method, that task can remain minimal. Indeed simply describing the lightweight QualOSS assessment method in a document is a sufficient mechanism to guarantee traceability. Evan a lightweight QualOSS assessment method is often created so it will be applicable for assessing several F/OSS endeavors. Thus if time permits, it is usually a good idea for the evaluator to conduct a debriefing after each assessment in order to record his impressions regarding how the assessment went and where mishaps eventually occurred.

5.2.2 Enabling lightweight QualOSS assessment

To conduct lightweight QualOSS assessment, it is important to leverage on existing assessment results from other middleweight or heavyweight QualOSS assessment method, for instance, results obtained from applying the standard QualOSS assessment method on F/OSS endeavors.

It is therefore our intend to publish the assessment results from the F/OSS endeavor analyzed during Work Package 3. These results are in the form of reports describing the full assessment and its final outcome. This means that it will not be simple to recompute adapted indicators if the lightweight QualOSS assessment indeed decides to adapt the indicators. Thus, we will also make available dumps of the assessment results in an electronic format suitable for machine processing. The current plan is to publish QualOSS assessment reports and database dumps on the QualOSS website http://www.qualoss.eu.

5.3 Heavyweight Applications of the QualOSS Methodology

In F/OSS acquisition context with very high stake, it may be warranted to request a type of assessment that address dimension not included in the standard QualOSS assessment method. For instance, characteristics that must be assessed are not found in the standard QualOSS assessment method or the measures and data analysis in the standard QualOSS assessment method are currently not thorough enough. Thus, additional measurements, data analysis and indicators must be developed to form a new advanced QualOSS assessment method. Creating an advanced QualOSS assessment method will require more time that creating and following a lightweight QualOSS assessment method and it will even take more time than following the standard QualOSS assessment method. In turn, we say that developing and then following new advanced QualOSS assessment method is an heavyweight approach.

However, a new advanced QualOSS assessment method needn't be created from scratch. It can reuse parts of other existing QualOSS assessment methods and adapt them to the new context. Currently, there only exist the standard QualOSS assessment method but if additional assessment methods are develop and respect the QualOSS methodology requirements then they may also be used as basis for create a new advanced QualOSS assessment method.

The heavyweight approach will be explored during the case studies of Work Package 5. In particular, Work Package 5 will determine how the standard QualOSS assessment method needs to be adapted to answer the concerns of two real life situations. Both real world scenarios targeted are described in deliverable D5.1.

The intend of Section 5.3 is to highlight how a QualOSS assessment method may be adapted in general. In other words, it explains how one may reuse part of existing QualOSS assessment methods and the techniques to use to adapt and then aggregate these parts into a new advanced QualOSS assessment method. This section, only list guidelines, workflows, rules (restrictions) and other methods and techniques useful for customizing existing methods or even for creating a new method and make sure that the resulting customized or new method stays within the expected spirit of the QualOSS methodology.

However, Section 5.3 does not present the actual work of adapting the standard QualOSS method to the real life situations. This effort will be explored during part of Work Package 5 and the results of adapting the standard QualOSS assessment method will be described in deliverable D5.3.

5.3.1 Constructing an Advanced QualOSS Assessment Method

Although one may create a QualOSS assessment method from scratch, it would be quicker and more reliable to construct a new QualOSS assessment method using existing ones, for instance, using the standard QualOSS assessment method as a basis and only adapting parts of it. Below, we review parts of a QualOSS assessment method that will most commonly require alterations.

It is assumed that the broad context, the justification for a new type of assessment have been clearly identified and described. Thus, the techniques presented below do not cover how to perform the task of initiating an assessment. Clearly, an company that plans to exploit the QualOSS methodology and to create new QualOSS assessment methods would clearly benefit from developing questionnaires and interview techniques to help with the task of initiating an assessment. For instance, develop questionnaires and interview techniques that will explicitly cover all the roles that have an interest in assessment results, will clearly identify the type of company projects that need to perform the assessments, etc.

Concerning the tasks of collecting and analyzing data and of interpreting results, those proposed in the standard QualOSS assessment method are fairly generic. The main difference may consists of launching a different tool than the QualOSS platform to collecting and to analyze data and then to interpret the results.

The supervision of an assessment is also a task performed generically in the standard QualOSS assessment method. Furthermore, all the supervision effort spread in the various workflow operations of standard QualOSS assessment method are required in order to meet the traceability requirement set by the QualOSS methodology. The only optional part of the supervision is the final debriefing at the end of assessments We believe however that it would be a step in the wrong direction to eliminate this debriefing. It takes little time and in the long term it is a simple instrument to improve an QualOSS assessment method.

The setting up and planning task can however significantly be tailored. First, a new QualOSS assessment method may adapt the set of rules or the workflow used in the standard QualOSS assessment method.

Second, new characteristics may be added to an existing quality model, for example, characteristics such as the compatibility of a F/OSS component or the geographical dispersion of community members. Even completely new quality model for top level characteristic may also be develop, for instance, define a quality model for the productivity of an F/OSS endeavor.

Third, new questions may be identified where answers will then be used to assess new or current characteristics of a quality model. For instance, to assess risks related to the geographical dispersion of community members, one may want to ask the question: “what is the proportion of F/OSS community members present on each continent?”

Fourth, new indicators for combining results of current or new data analysis and measurements may be defined. In turn, these new indicators may require new data analysis and measurements to provide answers to new questions or more accurate answers to existing questions. For instance, various analysis of the geographical distribution of various types of community members (developers, translators, users, documentation writer, ...) will need to be developed to answer the question above. Furthermore, new indicators specifying how to combine answer to these questions will be defined. These indicators will aim at computing assessment scores for geographical dispersion that address an enterprise's concern for a particular F/OSS acquisition context.

It is important that the final QualOSS methodology specifies what can be done and what cannot be done for the four types of alterations mentioned above. For the first alteration, that is, adapting the rules of the workflow, specifying what is allowed is straightforward. In particular, any modification to rules or the workflow are acceptable as long as the resulting QualOSS assessment method satisfies all the requirements set by the QualOSS methodology.

However for the second, third, and forth types of alterations, that is, modification to quality models, to questions, to data analysis and indicators, specifying what is allowed by the QualOSS methodology is not as simple. These questions are addressed by the validation strategy described in Section 6.3. In brief, it is important for the QualOSS methodology first to determine when a data analysis tool or procedure is reliable and accurate enough to be used in a QualOSS assessment method. Second, it is important also to give guidelines on how new indicators can be validated.

Once Task 4.4 has explored thoroughly the various kind of validation, we will then add new requirements to the QualOSS methodology regarding what new data analysis tools and procedures and what new indicators must satisfy in order to be used in QualOSS assessment methods.

6 Validation Plan

QualOSS performs four types of validations: (1) data validation, (2) data analysis validation, (3) usefulness validation, and (4) validation of the QualOSS platform.

Data validation is handled in WP3 in particular in Task 3.3. It consists of checking validity and completeness F/OSS data used for assessing an F/OSS endeavor. The strategy used for validating data is described in Section 6.1.

Data analysis validation is handled in WP4 in particular in Task 4.4. It is subdivided in two parts. First, data analysis validation consists of validating advanced metrics, that is, verify that the metrics proposed are indeed theoretically valid measurements. Second, data analysis validation consists of validating the indicators proposed. An indicator is a formula of metrics on which thresholds were specified in order to categorize results as inducing more or less risks.

Usefulness validation is handled in WP4 and WP5. In WP4, the usefulness of the method for interpreting the results is studied and in particular, the interpretation guide and other use manual for the QualOSS platform will be studied. On the other hand, WP5 studies the usefulness of the results obtain when applying the QualOSS methodology. It does so by conducting case studies on two different real-life situations.

Validation of the QualOSS platform is handled in WP2 in particular, Task 2.3. This verification task will check that the implementation of the platform respects the requirements and specifications. Furthermore, verifications of analysis tools prior to connecting them to the QualOSS platform are also presented as part of Task 2.3.

The strategy for validating the QualOSS platform and for validating the usefulness of the QualOSS methodology are well understood. The former merely consists in validating the implementation with respect to the requirements and specifications and the latter consists in conducting case studies in WP5. Therefore, below we further present the strategy for the other types of validations.

6.1 FLOSS Data Validation Strategy

Data validation deals with the validation of actual data and of measurements on this data. The validation is an iterative process, consisting of four stages. The first two rounds deals with *input to a QualOSS assessment* and the available data sources, the third stage with validating the *selected input* to an QualOSS assessment, and the fourth round with the *output* (or results) of a QualOSS assessment.

The target group that is to be interacted with for all the four stages are active actors from the open source communities around the relevant F/OSS endeavor being assessed. Potential F/OSS community members to contact for performing the validation are identified and listed in D3.1.

Each round is described in detail below, indicating the reason why the task is necessary, and what input is required for this task to be executed.

6.1.1 First round: Data sources – documentation and availability

First of all, the data needs to be available and its location must be known and recorded. For each F/OSS endeavor under assessment, it is necessary and paramount to properly record where the data comes from, and for each artifacts to indicate its exact source(s). Furthermore, it is of uttermost importance to annotate artifacts with additional information so analysis can be replicated or so results of the same analysis at different points in time can be compared. In consequence, for each artifact, one must be able to recover the version to which it belongs, the date when it was created, when its was measured, the exact source from

which it comes, and the share of the project actually measured (e.g. The tool may be able to deal with only a limited number of file types or programming languages, so for example only part of the code could be analyzed due to these limitations)

In other words, the first round checks that F/OSS data is available, to what extent it is available, and more importantly, that its source was properly recorded as well as other important attribute of the data. This stage does not require interaction with F/OSS communities, but is an important step merely to validate proper documentation of what data is used, coverage, and where and when it is obtained.

6.1.2 Second round: Data sources – reliability and representativeness

The second stage validates the extent to which the available data is reliable and/or complete for the F/OSS endeavor in question.

Depending on the source of the data, it is necessary to inquire whether the source is reliable and/or complete, for instance: Is the CVS tree used for a QualOSS assessment actually representative of the whole project? It is necessary to check every data source, and a validation strategy must be further elaborated for the various data sources and types of artifacts.

This stage in the iterative process will also allow for feedback concerning missing data. If available the missing data will be added to be considered by the QualOSS assessment. Otherwise the results of the QualOSS assessment will be annotated with the fact that they were obtained on incomplete datasets, giving an indication of the extent of coverage and completeness.

6.1.3 Third round: Selected sources - actual metrics

The third round deals with the selected inputs and the measures obtained from these inputs. The purpose of this third stage is to ask F/OSS community members if the actual measures seems to reflect reality as preceived by the FlOSS community members. There are at least two reasons for a measure to seem unrealistic. First, the data source is incomplete but the community member did not notice it in the previous round. It may be very overwhelming to validate large amount of data. On the other hand, it becomes easier to validate the completeness and accuracy of a dataset from measures obtained on it. A second reason for unrealistic measure is a flaw in the implementation of the measurement or analysis tool. It is important to have this continuous feedback in order to improve how the QualOSS methodology is applied.

6.1.4 Fourth round: Indicators and output

The fourth and final round is about validating the interpretations and conclusions obtained from aggregating measures, in particular, by checking the values of indicators and how the QualOSS assessment propose to interpret them. A similar validation is also perform in WP4 task 4.4. The difference being that in WP3, we check with F/OSS community members while Task 4.4 performs other kind of validations such as expert review or empirical.

The main purpose of this fourth round is to stimulate a dialog between F/OSS communities, FlOSS users and F/OSS researchers.

As mentioned above, this round requires interacting with F/OSS community members. We already observe that the opinion of F/OSS community members may be quite subjective. In turn, we view this round as a right granted to F/OSS community members to comment on the results of an assessment rather than a true validation. However, we do not neglect the case of a flawed indicator is flawed and in such cases, the comment by F/OSS community members should easily explain why the indicator value should be discounted in their cases. Obviously the purpose and goals of the actor wanting to perform of the assessment of a F/OSS endeavor may be different than for someone active in the community, and this should be kept in

mind, at the same time as checking to what extent something “makes sense”, and providing feedback to the community, and giving them the opportunity to participate in the process.

6.2 Validation of the QualOSS Platform

This tasks verifies if the QualOSS platform respects the requirements and specifications listed in deliverable D2.1. This validation task will actual take place in Task 2.3. In addition to verifying the proper implementation, user validation of the QualOSS platform will be performed.

6.2.1 Implementation Verification

The objective of this task it to verify that the QualOSS platform is implemented as specified in its requirements and specifications document (D2.1). Currently, it is anticipated that the following verifications will take place:

- Verify that the proper constraints were implemented in the QualOSS database schema,
- Verify that the connection mechanism to connect analysis tool to the QualOSS platform work as expected
- Verify that measures obtained from erroneous connectors can be cleaned from the QualOSS database
- Verify that indicators compute properly

6.2.2 User acceptance

First, we note that the typical user of the QualOSS platform is a technology expert. In other words, the QualOSS platform is not intended for use by lambda users. Furthermore, we observe that only CETIC has contributed to the implementation of the QualOSS platform, although other QualOSS partners contributed to the implementation of analysis tools and connectors to connect analysis tools to the QualOSS platform, they did not participate to the implementation of the QualOSS platform itself. In turn, our strategy for user validation is to ask other QualOSS partners such as URJC and AdaCore to deploy the QualOSS platform and to execute QualOSS assessment using the QualOSS platform and its documentation.

6.3 Validation of the QualOSS methodology

Deliverable D4.4 *validation of quality models and user manual* describes the different verification and validation activities that are to take place during task 4.4. At the time of proposal writing, the notion of indicators was not defined, instead the proposal used the terms metric formula and calibration. In turn, what the proposal referred to as validating calibration can now be equated to validation of indicators.

Furthermore, at the time of proposal writing, it was not anticipated that WP1 would conduct interview with FlOSS integrators. In turn, the validation of the quality models was already performed during WP1. The new quality models propose in D4.1 are already responds to the weaknesses in the quality model proposed in WP1 identified by the F/OSS integrators as well as the weaknesses identified during the Year-1 review of QualOSS. In turn, for the purpose of the QualOSS project, the quality models for the robustness and evolvability of F/OSS endeavor proposed in D4.1 are considered validated.

In consequence, the relevant validations to validate the QualOSS methodology are:

- Validate metrics use in indicators
- Validate indicators

The validation strategy for each of these validation is briefly described below.

6.3.1 Task 1 – Validation of metrics

Input: List of metrics used in indicators

Actions: Check the consistencies, correctness, completeness of metrics. Are those metrics built on sound theoretical foundation (units, scales, sound definitions...), and if needed, was a metric validated on a large enough set of real-world data?

6.3.2 Task 2 – Validation of indicators

Input: List of indicators use in our quality models

Actions: Check the validity of indicators

In order to validate indicators, we propose three strategies:

- Peer review
- Consistency Checking
- Empirical Validation

Practically, our quality models are hierarchy of quality characteristics which have been identified as influencing the robustness and evolvability of F/OSS endeavors. When applying the QualOSS methodology, one first identify questions that must be answered in order to assess the degree to which the characteristics is present in a F/OSS endeavor. The relevance of questions may vary according to the business context. In a second phase, indicators that help answer the questions are selected.

In the middleweight application of the QualOSS methodology presented in Section 5 of D4.1, a set of commonly asked questions and of indicators to answer these questions are proposed. In practice, an indicator is create by experts from the QualOSS consortium. A first validation strategy consists in making another experts from the QualOSS consortium review and accept each propose indicator.

Furthermore, in many cases, the questions to answer using indicators are high-level enough. In turn, several indicators can be defined to propose alternate ways to answer one question. It is then possible to propose a validation based on consistency checking. In particular, if the values of the different indicators tend to the same results then we may say that they are consistent and that they probably answer the question well. On the other hand, when their values are contradictory, it is important to have a reliable interpretation guide to help the user decide which indicators to listen to and which to ignore (see validation of the user manual below).

Finally, some indicators will be validated empirically. As it is true for all software development activities, F/OSS developments are human intensive effort. Due to the wide variation in human abilities and capabilities, most indicators are not likely universal, in particular, what is risky in one case is not in another. In consequence, we believe it is more appropriate to study the validity of indicators on case basis, that is, for each individual F/OSS endeavor independently. Only if certain indicators validate on several occasion is it then useful to conduct larger empirical studies on their validity. However, due to time consideration, such large empirical sutdies cannot take place during the QualOSS project. Indeed, QualOSS will only validate indicators at the level of individual F/OSS endeavor.

Furthermore, indicators that were not empirically studied must be open to criticism. In particular, it is important to open to public debate the indicators proposed in the standard QualOSS assessment method. In turn, industry and academic experts as well as F/OSS community members should have means to share their opinion with regards to standard QualOSS assessment method and in particular, on the indicators it

uses. Such comments will be collected as part of WP5 and in the round four of validation described earlier in Section 6.1.4.

6.3.3 Validation of the interpretation guide (or user manual)

The user manual will facilitate the understandability and the usage of the QualOSS methodology and in particular, it will explain how the values of indicators should be interpreted depending on the context of the FlOSS acquisition scenario. The proposed strategy for validating the user manual, including the interpretation guide (or rules) is expert and user review. In particular, we believe that WP5 (case studies) where the QualOSS methodology will be applied on two real-life situations proposes an adequate ground for validating the user manual and its interpretation rules.

Input: user manual (including the interpretation guide) in deliverable D4.2

Actions: Check the completeness and the understandability of the user manual and of its interpretation rules.

7 Summary and Conclusions

This deliverable initially explains that when acquiring F/OSS component, enterprises are not only interested in the quality of the product but also in the context surrounding a F/OSS endeavor including aspects related to a F/OSS community, the software processes followed and tools and dependencies on other F/OSS endeavors. Thus, when faced with an F/OSS acquisition decision, enterprises would like to assess all these aspects.

As a consequence, the QualOSS methodology propose a flexible and rigorous approach for assessing F/OSS endeavor. Furthermore, a standard QualOSS assessment method that respects all the requirements prescribed by the QualOSS methodology is developed. The standard QualOSS assessment method enable assessing F/OSS endeavor for an acquisition context, called the full F/OSS collaboration, where an enterprise plans to integrate a F/OSS component in a software product and also wants to keep the contribution channel open in both directions, that is, it wants to be able to contribute the F/OSS endeavor and also retrieve contributions from others if desired.

In the full F/OSS collaboration context, an important decision factor relates to the robustness and evolvability of an F/OSS endeavor respectively, its capacity to solve current problem and to last in the future. Furthermore, various types of employees in an enterprise have an interest in the assessment results, for instance, product managers, project managers, architects, analysts, developers, testers and even technical writers. In turn, the standard QualOSS assessment method answers questions of interest to each of these roles and evaluate assessment goals by aggregating answers to these questions. The assessed goals identify and evaluate the risks related to the robustness and evolvability of an F/OSS endeavor.

Once evaluators are familiar with the standard QualOSS assessment method, it is expected that assessing a F/OSS endeavor will require about 1 person-week of effort. In some cases, people may not have resources available to perform F/OSS assessment, we therefore present a lightweight QualOSS assessment method that explain how to exploit existing evaluation results with very little effort (around 1 hour). Conversely, in certain very high-stake context, more specific F/OSS assessments are needed. The heavyweight approach explains how new assessment methods can be derived from the standard QualOSS assessment method and maintain all the requirements set by the QualOSS methodology.

On a final note, we indent on keeping the QualOSS label as a synonymous of *rigor in assessment*. So if people do not find the standard QualOSS assessment method sufficient for their purpose and want to create new QualOSS assessment methods either by adapting the standard QualOSS assessment method or from scratch then it may be important to have a mechanism to verify that new assessment methods still respect the requirements prescribed by the QualOSS methodology hence deserve the QualOSS label. Acquiring such a label would be required if one wants to share F/OSS endeavor assessment results and claim they were obtained following a QualOSS assessment method. In turn, to be prepare in cases where others want to create new QualOSS assessment method, the QualOSS consortium already looks at how a QualOSS board would be created in order to assign QualOSS label. However, the actual creation of a QualOSS board will only take place if required.

REFERENCES

(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.

(Deprez *et al.*, 2007) Jean-Christophe Deprez, Frédéric Fleurial Monfils, Marcus Ciolkowski and Martin Sóto. *"Defining Software Evolvability from a Free/Open-Source Software Perspective"* in proceedings of the Third International IEEE Workshop on Software Evolvability 2007, Oct. 1st, 2007, Paris, France, pp. 29-35.

(Deprez, 2008) Jean-Christophe Deprez and Simon Alexandre, *"Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR & QSOS"* in the proceedings of the 9th International Conference on Product Focused Software Process Improvement (PROFESS 2008), Eds. Andreas Jedlitschka and Outi Salo, Monte Porzio Catone, Italy, June 23-25, 2008, pp 189-203.

(Henderson-Sellers, 2002) Brian Henderson-Sellers “Process Metamodelling and Process Construction: Examples Using the OPEN Process Framework (OPF)” Ann. Software Eng, Vol. 14, Number 1-4, pp. 341-362, 2002.

(Henderson-Sellers, 2005) Brian Henderson-Sellers and Cesar Gonzalez-Perez and M. K. Serour and Donald G. Firesmith, *“Method engineering and COTS evaluation”* ACM SIGSOFT Software Engineering Notes, 30(4), pp. 1-4, 2005.

(Bjorke, 2005) Bjørke, S.Å. (2005) 'The Concepts of Communities of Practice, Activity Theory and Implicationsfor Distributed Learning' in Lager, R. (ed) *Artikler om høgskolepedagogikk*, *Skriftserien nr 118*, Agder University College, Kristiansand 2005, ISBN: 82-7117-558-0

(de Souza, 2003) de Souza, C. R. B., Redmiles, D. F., Opportunities for Extending Activity Theory for Studying Collaborative Software Development, Workshop in Applying Activity Theory to CSCW research and practice, European Conference in CSCW (ECSCW 2003), Helsinki, Finland, September 14-18, 2003.

(QSOS, 2006) Method for Qualification and Selection of Open Source software (QSOS) version 1.6 © Atos Origin, April 2006 (http://qsos.org/).

(OpenBRR, 2005) Business Readiness Rating for Open Source © OpenBRR.org, BRR 2005 – Request fro Comment 1, 2005, (http://www.openbrr.org).

(FLOSSMETRICS) http://www.flossmetrics.org (Accessed in May 2008)

(FLOSSMole) http://ossmole.sourceforge.net/ (Accessed in May 2008)

(Ohloh) http://www.ohloh.net/ (Accessed in May 2008)

Appendix A: QualOSS Quality Model and Assessment Goals

This section presents the hierarchical quality model used in the standard QualOSS assessment method. First, Figure 6 shows that work products, community members, software processes and tools will be studied as part of an assessment. Second, the leaf quality characteristics translate into assessment goals. As explained below, we use an approach based on the Goal-Question-Metric paradigm (GQM) where assessment goals are derived from characteristic definitions combined with the full F/OSS collaboration acquisition context described in Section 3.2.1.2. The quality model of Figure 6 is skinnier that the one presented in Figure 2. We eliminated quality characteristics that are not of high interest in the full F/OSS collaboration context or characteristics that cannot be evaluated in a generic way, for instance, compatibility or operability can only be assessed if the target environment and users are known without such an information, assessment results would be to general hence useless.

GQM proposes a prove approach for performing guided assessment with particular business goal in mind. Indeed, the first step of this paradigm suggests to start from business goals and then derive assessment goals. In QualOSS, we decided to start from the full F/OSS collaboration scenario. The business goals are then to identify the risks related to the integration of a F/OSS component in a software product. An enterprise wants to be able to contribute to the F/OSS endeavor and also to retrieve contributions from others. Risks should not be identified and evaluated strictly from a technical view point but also from the view point of management whose concerns are more related to the predictability of the F/OSS endeavor's behavior. Furthermore, once a F/OSS component has been integrated, in the full F/OSS collaboration, the enterprise wants to maintain and even augment its partnership with the F/OSS endeavor that is producing the F/OSS component.

The overall risks of integrating a F/OSS component in a software product in the full FlOSS collaboration context are partly influenced by risks related to the robustness and evolvability of an F/OSS endeavor, that is, the respective capability of the F/OSS endeavor to solve current problems and to last in the future. The top level assessment goal of the standard QualOSS assessment model is therefore to evaluate the *risks related robustness and evolvability of an FlOSS endeavor*.

Top-level Assessment goal

Definitions of Robustness and Evolvability of an FlOSS endeavor:

- The ***robustness of an FlOSS endeavor*** is the degree to which an F/OSS endeavor is capable to keep functioning when mishaps occur —a mishap may be internal or external to the F/OSS endeavor in question. For example, a bug being reported or a dispute among community leaders are internal mishaps. A technological shift or the appearance of a new competing F/OSS endeavor are examples of external mishaps.
- The ***evolvability of an FlOSS endeavor*** is the degree to which a F/OSS endeavor is capable to remain viable in the long future.

Assessment Goal for the robustness and evolvability of an FlOSS endeavor:

- Evaluate the degree of risk related to the robustness and evolvability of a F/OSS endeavor for an enterprise in front of the full FlOSS collaboration acquisition scenario. Risks may be evaluated based on the current stated of the F/OSS endeavor, on the past experiences of the F/OSS endeavor on robustness and evolvability issues, and on the evolution trend of how the F/OSS endeavor handles its robustness and evolvability.

Directly evaluating risks related robustness and evolvability of an FlOSS endeavor is too complex. In turn, the top-level assessment goal is broken down into simple assessment (sub-)goals. Figure 6 shows the quality model used in the standard QualOSS assessment method. It first identifies the four elements that constitute an F/OSS endeavor and then presents the characteristics of these elements that bear on the robustness and evolvability of an F/OSS endeavor.

This new advanced quality model solves the weaknesses of the prototype quality model identified during the validation task of WP1, Task 1.6. In particular, in Task 1.6, no acquisition scenario was specified. This made the assessment goals too broad and therefore the measurement and indicators were also broad. In turn, it was hard to understand what they were trying to evaluate. In Task 4.1, this problem is solved first the assessment (sub-)goals are not only associated to the definitions of characteristics but take into account a particular F/OSS acquisition scenario, namely, the full F/OSS collaboration scenario.

From the assessment goals, the next step of the GQM is to raise questions. The answers to these questions should be based on sound measures or sound data analysis. Furthermore, it should be possible to combine answers to these questions to define risk indicators.

The whole operation of identifying, combining measurement and specifying threshold on these aggregation of measurements is what we have called calibration. It is performed as part of Task 4.2 and the resulting indicators will be presented in deliverable D4.2.

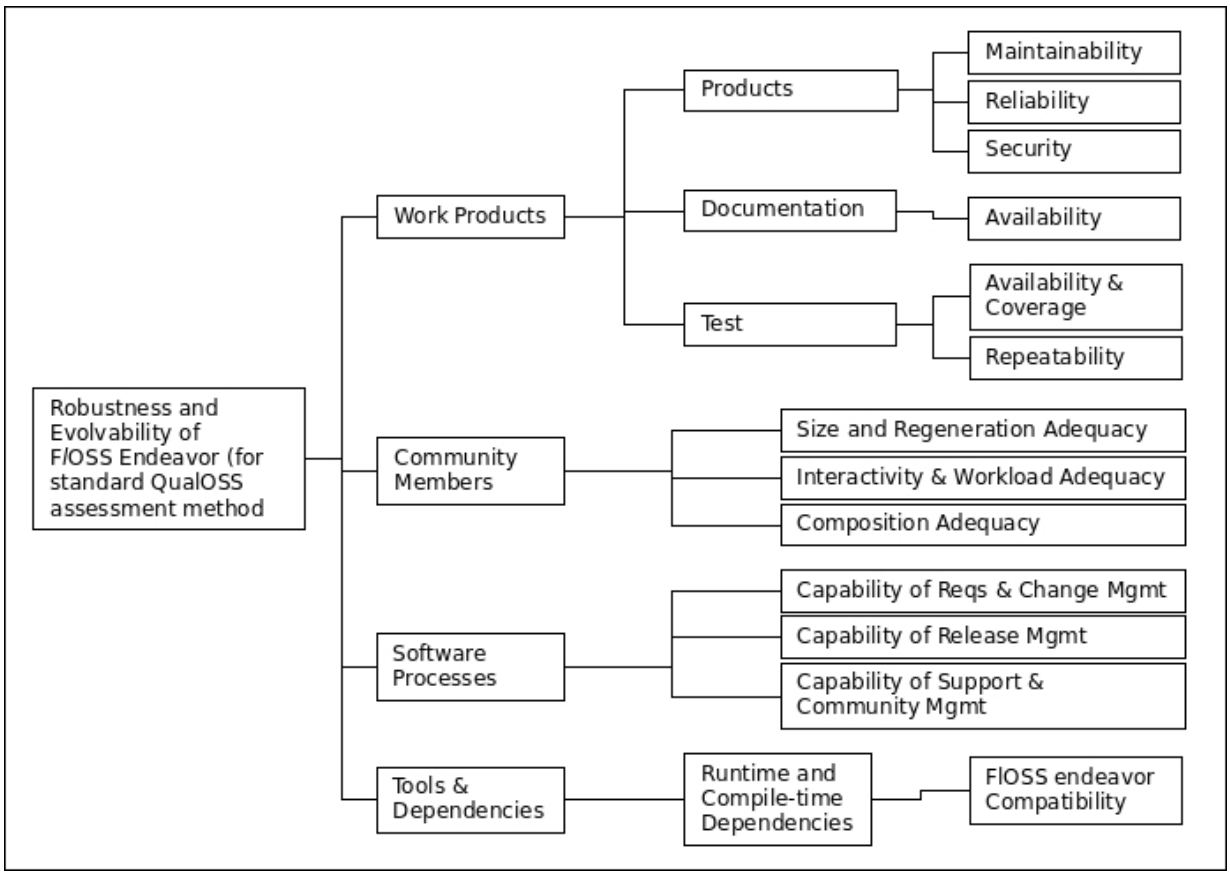

Figure 6: Quality model imposed by the standard QualOSS Assessment method.

Prior to presenting the definitions, assessment goals and questions related to the leaf characteristics of the quality model in Figure 6, we briefly recall that in the full F/OSS collaboration, people with various roles in an enterprise are interested in the assessment results of an F/OSS endeavor.

General Context of the Full FLOSS Collaboration and its viewpoints

As previously mentioned in Section 5.1.2.1, in the full FlOSS collaboration context, the product manager, the project manager, architects, analysts, developers, testers and eventually, technical writer are interested in the results of an F/OSS endeavor assessment. Each role expresses a varying degree of interest in each characteristic, and even for a given characteristic, each role may have a slightly different viewpoint on the characteristic. In turn, each viewpoint asks different kinds of questions. Below, we briefly review the general interest that various roles have when faced with the full F/OSS collaboration scenario.

It is important to realize that in order to exploit the complete potential of a full F/OSS collaboration, the traditional role of product manager, project manager, architect, analyst, developer, tester and even technical writer must be broaden as compared to their respective roles when faced with the acquisition of a COTS component. Indeed, in the F/OSS world, many type of data are available that are usually concealed by traditional enterprises. Accordingly, these various roles must learn how to exploit the new data available in the F/OSS context. This will enable them to make more informed decision, i.e., decision based on real evidence against the traditional decision making based on product marketing plus the history and economical strength of the COTS vendor.

Viewpoint of Product Manager

Once the F/OSS component has been integrated in the software product and the product is ready for distribution, it will be under the management of a product manager. Prior to the development of the product, this manager would already like to present an initial ROI estimate for the software product to the higher-level management of an enterprise. Furthermore, the product manager has a long term view in mind. In turn, he is most likely interested to know about the evolution trend of the robustness and evolvability of a F/OSS endeavor rather than to know about the current state of an F/OSS endeavor.

He also wants to know that future contributions to the F/OSS component will be accepted, mainly to keep the F/OSS component in-line with the software product in which it has been integrated. The product manager will also be interested to know how long old releases are supported to plan F/OSS component upgrade appropriately. Besides, the product manager will also be concern with the rapidity with which innovation is introduced in the F/OSS component. Issues of licenses of the F/OSS components will also be addressed by the product manager. One important question regards the coherence between the licenses used in the various code files of the F/OSS component and the actual license under which the overall F/OSS component is released.

In summary, the product manager is less focused on the development project that integrate a F/OSS components in the software product and more on the future of the software product and how it will stay in tune with the F/OSS endeavor. However, he realizes that knowing information about the future is unfeasible but he usually wants to take his decision based on the historical evolution of how the F/OSS endeavor behaved in the past and how it made its F/OSS component evolve.

Viewpoint of Project Manager

The full FlOSS collaboration context assumes that a software product considers integrating a FlOSS component. This software product must take place under a software development project. That software development project also has a project manager (which could be the same or a different person from the product manager).

The main interest of the project manager is to keep the software project on track and on time. Thus, he is mostly concern about the robustness of the F/OSS endeavor, that is, its ability to solve current problems (in the F/OSS component and elsewhere.) Furthermore, the project management will be interested to know about F/OSS endeavor's release management approach. This will help determine what version is stable

enough to be integrated in the software product. The project manager also wants to ensure that patches and new minor version of the F/OSS component can eventually be upgrade in the software product in the course of the software project without much effect on the software product. Concerning support, in the case of blocking issues with the F/OSS component, the project manager also needs to know about support availability, even commercial support.

Viewpoint of Architect, Analyst and Developer

Architects, Analysts and Developers directly interact with the F/OSS component in the actual version of the F/OSS components to integrate in the software product. Their interest mainly has a technical dimension. Their task of integrating the F/OSS component in the software product will be impacted by the complexity of the code, which can be alleviated by a modular implementation with limited coupling between modules. Furthermore, complexity is easier to handle if technical documentation is available, and when not, if technical questions find consistently quick and acceptable responses from the FlOSS community. Furthermore in the long term, they will be the ones who will explicitly contribute to the F/OSS endeavor. Thus, they will need to know and respect the software processes of the F/OSS endeavor. Finally, the architect role has an added dimension, that is, to serve as the technical expert who is able to translate the long term vision of the F/OSS endeavor to the product manager. In concert, the architect and the product manager will plan future contribution to the F/OSS endeavor. Thus, the architect will also have interest in looking at past evolution of the code of the F/OSS component in order to understand why the software architecture is as it is and better anticipate evolution and future effort of refactoring or rewrite.

Viewpoint of Tester

Depending how thoroughly the F/OSS component was tested and testing report have been made available, a tester will need to invest additional testing effort on the F/OSS component prior to an enterprise integrating it in a software product. For instance, if unit-test reports exist for the F/OSS component then testers will not have to conduct these test again. Testers will also be interested in leveraging on existing integration or regression tests so they can use them as basis to create test for the software product that integrates the F/OSS component and they will not have to conceive new test cases from scratch.

Testers will also want to know if batch automated testing is possible for the various unit, integration or system in regression test suites. This will simplify their task of re-testing the component in the case where company developers perform minor modifications to the FlOSS components in order to integrate it properly with the software product.

Viewpoint of Technical Writer

The final software product may have to describe functionality that are implemented in the F/OSS component. Thus, technical writer wants to leverage on existing documentation.

Low-level Assessment Goals and Questions for Quality Characteristics

Below, each characteristic of the quality model is presented along with its definition, its related assessment goal and how this goal can be assessed from the viewpoint of each role.

1. Work Products – Product – Maintainability

Definition: The degree to which the software product can be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.

Assessment goals for Product Maintainability Indicators:

- Evaluate the degree of risk related to the maintainability of a F/OSS component based on the current state of the F/OSS component, the evolution of the FlOSS component and the past experiences of the F/OSS endeavor with maintainability issues on the F/OSS component.

Interest and Questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- The rapidity with which enhancements are implemented
- The code stability of the F/OSS component over major releases, in particular regarding the public interfaces of the F/OSS components
- Previous large refactoring or re-writing efforts undertaken by the F/OSS endeavor and if planned milestones were respected
- The volumetric evolution of the code of various releases of the F/OSS components over time (in chronological order)

Measurement Questions:
- WP-Ma-1: What is the percentages of enhancements proposal that get accepted?
- WP-Ma-2: What is the rapidity with which accepted enhancements are implemented?
- WP-Ma-3: What is the percentage of changes in the code between major releases?
- WP-Ma-4: What is the percentage of changes to public interfaces in the code (external API) between major releases?
- WP-Ma-5: What is the evolution in code *volumetry* between various releases of the code over time (in chronological order)?

Remark to consider when creating Indicators:
- When enhancement categories (major, minor, ...) exist, create various indicators for acceptance and rapidity for each enhancement category separately and then an indicator for all combined categories (or just relevant subsets of categories)
- *changes to code* can be measured at the level of package, module, file, class, function or line
- *public interfaces* : if the F/OSS endeavor does not have a strategy to indicate properly its external interfaces then percentage = NULL
- *evolution in volumetry of the code* can be take on modules (packages, modules, namespaces), on files, on classes, on functions and on lines of code. Furthermore, the rate of evolution of code documentation can also be taken

Project Manager:

Viewpoint: The product manager is interested to know about
- The rapidity with which bugs are corrected in the version of the F/OSS component considered for integration?

- The code stability of the F/OSS component in the version considered for integration (in the consider patches) and in its past minor releases
- The dispersion of code changes when correcting bugs

Measurement Questions:
- WP-Ma-6: What is the percentage of bugs reported and not assigned (or whose resolution status is also not assigned)?
- WP-Ma-7: What is the rapidity with which bugs are corrected?
- WP-Ma-8: What is the percentage of changes in the code between minor releases?
- WP-Ma-9: How many patches have been submitted for the actual version of the F/OSS component considered for integration?
- WP-Ma-10: What is the percentage of changes to public interfaces in the code (external API) between minor releases?

Remark to consider when creating Indicators:
- When bug categories (major, minor, ...) exist, create various indicators for acceptance and rapidity for each category separately and then an indicator for all combined (just a relevant subset of categories).
- Distinction measurements on bugs that have been assigned vs. merely reported but not assigned should also be done.
- *changes to code* can be measured at the level of package, module, file, class, function or lines
- *public interfaces* : if the F/OSS endeavor does not have a strategy to indicate properly its external interfaces then percentage = NULL

Architect, Analyst and Developer:

Viewpoint: Architects, analysts and developers are interested to know about
- The modularity of the F/OSS component (architects are more interested about high-level modules such as package while for developers is more concerned with files, classes and methods)
- The complexity of the code including coupling, cohesion (at various level of the code), Halstead software metrics
- The technical documentation inside and outside of the code, including extensive documentation of public interfaces containing code samples
- The complexity evolution of the code of various releases of the F/OSS components over time (in chronological order) (interesting for architects)

Measurement Questions:
- WP-Ma-11: What are the high-level modules (package, module, namespace) in the version of the F/OSS component considered for integration? What is the coupling of each high-level modules?
- WP-Ma-12: What are the low-level modules (file, class) in the version of the F/OSS component considered for integration? What is the coupling of each high-level modules?
- WP-Ma-13: What are the measures on various code element in the version of the F/OSS component considered for integration for the measurement: number of lines of code, cyclomatic complexity, and efferent and afferent couplings)?
- WP-Ma-14: What is the percentage of code documentation in the version of the F/OSS component considered for integration?
- WP-Ma-15: What is the complexity evolution of various releases of the code over time (in chronological order)?

Remark to consider when creating Indicators:
- When bug categories (major, minor, ...) exist, create various indicators for acceptance and rapidity for each category separately and then an indicator for all combined (just a relevant subset of categories)
- *changes to code* can be measured at the level of package, module, file, class, function or lines
- *public interfaces* : if the F/OSS endeavor does not have a strategy to indicate properly its external interfaces then percentage = NULL

- *complexity evolution of the code:* For example, the coupling between modules can be taken on modules (packages, modules, namespaces) and then summed up to observe its evolution over time, the same can be repeated on files and on classes.

2. Work Products – Product – Reliability

Definition: The degree to which a F/OSS component product can maintain a specified level of performance when used under specified conditions.

- NOTE: several measurement question request bug count. This is simple to obtain from most bug tracking system. However, it is much harder to perform the needed measurements on other type of repositories (such as email archive, unless emails are structured and coming from a bug tracking system). In the full F/OSS collaboration scenario, it is unlikely that an enterprise considers a collaboration with an F/OSS endeavor that does not offer and actively use a issue/bug tracking system. As a consequence, the absence of such a system in an F/OSS endeavor may automatically result in poor indicator values for the corresponding questions.

Assessment goals for Product Reliability Indicators:

- Evaluate the degree of risk related to the reliability of a F/OSS component based on the current state of the F/OSS endeavor and the past experiences of the FlOSS endeavor on reliability.

Interest and Questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- Numbers of bugs in each version of the F/OSS component labelled as *stable*
- History of bugs evolution (and stabilization) in minor releases within in the same major releases of the F/OSS component.

Measurement Questions:
- WP-Re-1: What is the number of bugs in each version of the F/OSS component label stable?
- WP-Re-2: Is there a decrease in the number of bugs reported in the life time of every major releases?
- WP-Re-3: After how many minor releases does the F/OSS component in a major release has a significant decrease in its bug count (from reporting)?

Remark to consider when creating Indicators:
- Various indicators can account for the different categories of bugs (severity level)
- In the absence of an issue/bug tracking system, the answer to question may be set to NULL.

Project Manager:

Viewpoint: The product manager is interested to know about
- Past bugs and their history for the version of the F/OSS component considered for integration only for minor releases
- Complexity of the code where bug corrections where needed in prior minor releases and in current patches (the more complex is the code where a bug corrections is needed the more chances there are to create new bugs)

Measurement Questions:
- WP-Re-4: What is the number of bugs in each version of the F/OSS component labeled stable in the actual major release of the F/OSS component considered for integration?
- WP-Re-5: Is there a decrease in the number of bugs reported in the life time of the actual major release of the F/OSS component considered for integration?
- WP-Re-6: What are the sums of the coupling for files or classes modified during bug corrections for each previous minor release (or eventually of patches) within the actual major release of the F/OSS component considered for integration?

Remark to consider when creating Indicators:
- When bug categories (major, minor, ...) exist, an indicator may be create for each category
- Bugs whose resolution status are DUPLICATE should not be counted

Architect, Analyst and Developer:

Viewpoint: Architects, analysts and developers are interested to know about

- Violation to code convention recognized for being risk to induce bugs (in the present or the future). The logic is that the enterprise may ask its developers to correct the F/OSS component code so it respect such convention. In turn, a high number of violation count will require a significant effort.

Measurement Questions:

- WP-Re-7: How many violations to industry convention are there in the current release of the FlOSS component considered for integration?

Remark to consider when creating Indicators:

- Tools that check the style of code usually have different level of priority to assign to a convention, a different indicator may be defined for each of the priority. Eventually some priority may be ignore and sums of violation for certain subset of priority may also be useful as indicators

3. Work Products – Product – Security

Definition: The protection of system items from accidental or malicious access, use, modification, destruction, or disclosure. [ISO/IEC 15026:1998]

- NOTE: a renown project that categories experienced vulnerabilities exist. It is call Common Vulnerability and Exposure (CVE). It is quite a comprehensive list of vulnerabilities that were experienced by people with certain software component. It hosted by MITRE. A more convenient way to search for CVE entries is through the National Vulnerability Database provided by NIST (nvd.nist.gov)

Assessment goals for Product Security Indicators:

- Evaluate the degree of risk related to the security of a F/OSS component based on the current state of the F/OSS endeavor and past experiences of the F/OSS endeavor on security.

Interest and Questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- History of security vulnerabilities that people using any version of the F/OSS component experienced.

Measurement Questions:
- WP-Sec-1: What is the number of entries in the National Vulnerability Database for all releases of the F/OSS component considered for integration?

Remark to consider when creating Indicators:
- Various indicators can account for the different severity level assigned to security vulnerabilities (high, medium and low)

Project Manager:

Viewpoint: The product manager is interested to know about
- History of security vulnerabilities that people use the F/OSS component experienced.

Measurement Questions:
- WP-Sec-2: What is the number of entries in the National Vulnerability Database for the current major release of the F/OSS component considered for integration?
- WP-Se-2: What is the number of entries in the National Vulnerability Database for the exact release of the F/OSS component considered for integration?

Remark to consider when creating Indicators:
- Various indicators can account for the different severity level assigned to security vulnerabilities (high, medium and low)

4. Work Products – Documentation – Availability

Definition: The degree to which various types of documentation are available.

Assessment goals for Documentation Availability Indicators:

- Evaluate the degree of risk related to the availability of documentation on a F/OSS component based on the current state of documentation available generated by the F/OSS endeavor and on the history of documentation availability throughout the lifetime of the F/OSS endeavor and the different versions of the F/OSS components.

Assessment goals and questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- Availability of various kinds of documentation on the latest release of the F/OSS component of interest.
- Availability of various kinds of documentation for older major releases of the F/OSS component of interest.

Measurement Questions:
- WP-Doc-1: Are various kind of documentation available with the most recent release of a F/OSS component?
- WP-Doc-2: Have the various kind of documentation been consistently available with older major releases of the F/OSS component?

Remark to consider when creating Indicators:
- Kind of documentation to consider may vary according to context but here are some common documentation type: user manual (or user guide) including tutorials and quick start guide, reference manual (list of user functions), API documentation, code documentation, installation guide, configuration guide, change log between releases,

Project Manager:

Viewpoint: The product manager is interested to know about
- Availability technical documentation for the actual version of the F/OSS component considered for integration. Technical documentation are API documentation, code documentation, advanced configuration documentation, architecture documentation, etc.

Measurement Questions:
- WP-Doc-3: Are various kind of technical documentation available for the desired version of the F/OSS component.

Remark to consider when creating Indicators:
- Beside common technical documentation mentioned above, other knowledge base in less structured format such as web forum may also be considered of interest

Architect, Analyst, Developer, Tester and Technical Writer:

Viewpoint: The architect, analyst or developer is interested to know about
- same interest as the project manager

5. Work Products – Test – Availability & Coverage

Definition: The degree to which various test suites (unit, integration, functional, system) exist and cover comprehensively the FlOSS component under test (functionally as well as in term of code coverage)

Assessment goals for Test Availability & Coverage Indicators:

- Evaluate the degree of risk related to the availability and coverage of test suites based on the the availability and coverage of test suites for the desired version of a F/OSS component and for other versions the F/OSS component through the life of the F/OSS endeavor.

Interest and Questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- History of test reporting

Measurement Questions:
- WP-Test-Av-1: Are test reports published for each releases of the F/OSS component?

Remark to consider when creating Indicators:
-

Project Manager:

Viewpoint: The product manager is interested to know about
- Various test reports for the desired version of the F/OSS endeavor

Measurement Questions:
- WP-Test-Av-2: Are test reports published for the desired version of the F/OSS component?
- WP-Test-Av-3: Do test reports show that tests covered the F/OSS component extensively?

Remark to consider when creating Indicators:
- Coverage measures may be taken on user functions or on code (file/class coverage, function/method coverage, lines of code coverage)

Architect, Analyst, Developer and Testers:

Viewpoint: Architects, analysts and developers are interested to know about
- Availability of test suites for the desired version of the F/OSS component

Measurement Questions:
- WP-Test-Av-4: Are test suites available from the F/OSS endeavor?

Remark to consider when creating Indicators:
- Various kind of test suites may be considered, for instance, unit test suite, user-function test suite, etc.

6. Work Products – Test – Repeatability

Definition: The degree of ease with which tests may be repeatedly launched on a F/OSS component.

Assessment goals for Test Repeatability Indicators:

- Evaluate the degree of risk related to the repeatability of various tests based on the repeatability of test for the desired version of a F/OSS component and through the life of the F/OSS endeavor.

Interest and Questions for the various viewpoints

Tester:

Viewpoint: Architects, analysts and developers are interested to know about
- Availability of test scripts for (re-)launching various type of testing (unit, integration, functional)
- Availability of documentation on how to conduct tests on the desired version of the F/OSS component

Measurement Questions:
- WP-Test-Rep-1: Are test script for re-running tests available?
- WP-Test-Rep-2: Is there documentation on how to re-run tests on the desired version of the F/OSS component ?

7. Community Members – Size and Regeneration Adequacy

Definition: The degree to which the size evolution and regeneration of a F/OSS community happens at an adequate rate to maintain a sustainable community size.

Assessment goals for Regeneration Adequacy Indicators:

- Evaluate the degree of risk related to the regeneration adequacy of a F/OSS community based on the historical evolution of regeneration through the life of an FlOSS endeavor.

Interest and Questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- Evolution of new bug reporting members, for instances over three-month periods
- Evolution of new code contributing members over three-month periods
- Evolution of new members contributing data other than bug reports or code over three-month periods
- Evolution of new core contributing members to the F/OSS endeavor, for instances over three-month period (core contributing members are members who contribute code frequently)
- Evolution of core contributing members who have stop contributing for instance, in the past six months
- Evolution of code committers still working on each previous major release of the F/OSS component (being supported)?
- Average longevity of committers to the F/OSS endeavor

Measurement Questions:
- Has the evolution of new community members reporting bugs remained stable or grown over the history of the FlOSS endeavor? (at least shown a stable or positive trend overall)
- Has the evolution of new code contributing members remained stable or grown over the history of the F/OSS endeavor? (at least shown a stable or positive trend overall)
- Has the evolution of new members contributing data other than code or bug report remained stable or grown over the history of the F/OSS endeavor? (at least shown a stable or positive trend overall)
- Has the evolution of new core contributing members remained stable or grown over the history of the F/OSS endeavor? (at least shown a stable or positive trend overall) (a core member is one with commit right who perform commits frequently for instance, more than once every three month period)
- What is the evolution of core members who stopped contributing for a significant period?
- Has the evolution of core members who stopped contributing for a significant period been compensated by the joining of new core members around the same time frame?
- What is the average longevity of committers to the F/OSS endeavor

Remark to consider when creating Indicators:
-

Project Manager:

Viewpoint: The product manager is interested to know about
- Number of code contributors who patched or committed changes to the desired version of the F/OSS component

Measurement Questions:
- What is the evolution of the number of code contributors who submitted patches or committed changes in the major release as the desired F/OSS component?

- What is the overall number of code contributors who submitted patches or committed changes in the major release as the desired F/OSS component?

Remark to consider when creating Indicators:

-

8. Community Members – Interactivity and Workload Adequacy

Definition: The degree to which the community interacts adequately and partition the workload among F/OSS community members adequately to maintain a community cohesion and motivation.

- NOTE: establishing appropriate social network to circulate information across community subgroups is considered under interactivity

Assessment goals for Reliability Indicators:

- Evaluate the degree of risk related to the interactivity and workload adequacy of a F/OSS community based on the historical evolution of F/OSS community interaction and workload sharing throughout the live of an F/OSS endeavor.

Interest and Questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- Rate of number of events (commits, issue reports, emails, forum posts) occurring every three-month period
- Rate of code commits occurring every three month period
- Rate of events vs number of community members who contributed them (workload per contributor)
- Average number events treated per contributing member over time
- Degree of contribution still happening for older version of the F/OSS component
- The completeness of the communication channel

Measurement Questions:
- Is the number of events adequate (to show a lively community)?
- Is the number of code commits adequate (to show a lively committer community)?
- Has the size of community supporting older versions of a F/OSS component remained sufficient? (compared to number of bug reports, the number of report fix vs those open and compared to code size)
- Are they sub-groups in the community? If so, are they disconnected or are active community members serving as bridges between these sub groups (sub groups can be at the level of roles such bug reporter, committer but subgroups can also be studied at the code level)

Remark to consider when creating Indicators:
- Indicators on the various type of community membership can be used to answer the question above. eg. contributor (= people who have contributed to any repository in the scope of the F/OSS endeavor), committer (= people who have committed information in a version control system in the scope of the F/OSS endeavor)

Project Manager:

Viewpoint: The product manager is interested to know about
- The size of the various community roles (code committer, issue reporter, translator, tester, ...) still working on the desired major release of the F/OSS component.
- Responsibilities assigned to code committers for the source code of the desired version
- Completeness of the know-how of the active community on the source code of the desired version of the F/OSS component?

Measurement Questions:
- Is the size of the community supporting the desired version of the F/OSS component sufficient? (for instance compared to number of bugs, correction rate and code size)

- Are the responsibilities of committers reasonable for the desired version of the F/OSS component? (for instance, if the source code is large then committer only commit changes in a portion of the code. Furthermore, are committers not assigned the same files at the same time?)
- Are current active code committers knowledgeable on the entire source code of the desired version? (eg. What is percentage of code files in the desired version of the FlOSS component where no contributions happened in the current and past major releases and where the committer who committed those files has not committed in a long period for instance, 1 year?)

Remark to consider when creating Indicators:

9. Community Members – Composition Adequacy

Definition: The degree to which the F/OSS community is composed of various adequate F/OSS community members to maintain the level of cohesion and motivation.

Assessment goals for Composition Adequacy Indicators:

- Evaluate the degree of risk related to the composition adequacy of a F/OSS community based on the historical variation of community composition attribute such as its heterogeneity, its redundancy

Interest and Questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- Companies whose employees participate to the F/OSS endeavor
- Companies whose people have leadership roles in the F/OSS community
- Companies providing services based on the F/OSS component
- Number of roles identified in the community (bug report contributors, committers, code contributors, other types of contributors such as testers, translators + eventually leadership roles such as release manager, project leader, ...)
- Redundancy of code contributors

Measurement Questions:
- What companies have contributed bug reports?
- What companies have contributed code (code patches or code commits)?
- What companies have employees in leadership position in the F/OSS community (project leader, release manager, treasurer, ...)?
- What companies provide services on the F/OSS component?
- What are the roles who are filled by active community members?
    - Are these roles adequate for the F/OSS endeavor in question?
- Is there a sufficient number of code expert on the various portion of the F/OSS component?

Remark to consider when creating Indicators:
-

10.Software Processes – Capability of Requirements and Change Management

Definition: The degree of capability with which an F/OSS endeavor handles its requirement and change management activities

- NOTE: An F/OSS endeavor is considered more capable to manage requirements and changes if it explicitly proposes a procedure to follow for proposing, deciding important enhancement and if a trace of requirement and change management is recorded. It is also more capable to manage requirements and changes if the procedure protect the F/OSS endeavor for unanticipated impact of changes.

Assessment goals for Indicators of Capability of Requirements and Change Management:

- Evaluate the degree of risk related to the capability of an F/OSS endeavor to handle requirements and change management process based on explicit procedures and past evidences of requirements and change management activities.

Interest and Questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- The platform used to discuss the long term evolution of an F/OSS component and other aspects of the F/OSS endeavor
- The procedure(s) to follow to propose enhancement to the F/OSS product and get them accepted
- The transparency of decisions regarding priorities for the future development of a F/OSS component endeavor
- The kind of impact analysis performed by community members to take decision on whether to integrate an enhancement proposal
- The decision procedure regarding re-write or large refactoring decisions
- The method to gain write access to the F/OSS endeavor's a version control system (or the chances that write access be granted one day to external contributors)
- The responsibilities and access assigned to committers
- The change management procedure to maintain a connection with external renown sources such as CVE
- Spreading corrections to other releases/versions of a F/OSS component
- The maintenance of the issue reporting system

Measurement Questions:
- Is there a platform or mechanism in place to discuss the long term evolution to the F/OSS component (including discussion on dependencies to other FlOSS components, on large re-write effort and refactoring)?
- Is there a platform or mechanism in place to discuss the long term evolution to the software processes and the process automation tools and their configuration?
- Does there exist a procedure (and eventually a template to fill) to propose a significant enhancement to a F/OSS component?
    - Does this procedure enforce to justify the decision? (whether accepted or rejected)
    - Does the procedure enforce that an impact analysis be performed (even if high level) and be presented as part of the justification of a decision?
- Are committer's responsibilities explicit? (e.g. verification of convention following in code, etc.)
    - Are they enforced by the version control system?
    - Are these responsibilities realistic? (e.g. number files to supervise under a acceptable threshold)
- Is there a procedure that explains how to earn commit rights (that is write access to the version control system and part of its tree)?
- Is there an effort to maintain links between issues (in the bug tracking system) and the Common Vulnerability and Exposure dictionary/database?

- Is there a known, adequate mechanism (and possibly a documented procedure regarding the patching of several supported releases with accepted corrections?
- Is someone assigned the role of maintaining the bug tracking system? (eg. dump issues from not unsupported versions, review long live tickets to make their status evolve, etc.)

Remark to consider when creating Indicators:

- Indicators will be based on categories of questions where each category represent a level of capability.
- An evaluator will then start answering questions in the category corresponding to the lowest capability. If a certain threshold of answers are positive, he then proceeds to the next category. When too many questions are answered negatively for a particular category then the capability level of the previous level is assigned
- Categories needn't differentiate between viewpoint of roles. This means that questions for viewpoints of product manager, project manager, architect, developer, analyst, testers and technical writers may be found in 1 single category for a given capability level
- Answers may not always be binary in turn, it will also be important to know how to convert a non-binary answer to a final yes-no answer. (e.g. some question may find a procedure or a trend is present but not always respected)
- If needed, questions may be annotated as mandatory yes (in order to be assign the corresponding capability level)
- Other software processes will follow the method to assign capability level

Project Manager:

Viewpoint: The product manager is interested to know about

- The kind of information to provide when reporting bugs in particular if special type of information are needed.
- The procedure for properly reporting an issue to the F/OSS component (search for prior occurrence, assigning severity level to bugs, ...)
- The procedure used by community members to assign resolution status of “will not fix”, “fix in a later version”
- The procedures to follow for corrective changes (e.g who can create and propose patches? or what is the chance that a patch be integrated into the code base line once well tested?)
- The (test) verification procedure that one must follow when proposing a corrective change

Measurement Questions:

- Is a bug/issue tracking system in use? else is there an appropriate mailing list of forum for reporting issues about a F/OSS component?
    - If not, is there a way to notify someone of problems with the FlOSS component (email, forum)?
- Is there a procedure and a template of information to provide for proper issue reporting?
- Are issues assigned a responsible person and status within an acceptable delay?
- Are the unfortunate status of “will not fix” or “fix in a later version” explained to the issue reporter?
- Are issues transiting from a closed status to reopen limited to a few cases?
- Is there a known mechanism (or documented procedures) in place to propose corrections (eg. test change, submit a patch with such and such info, etc.)
    - Are the verification requirements for corrections explicit and reasonable?
- Have corrections from people without commit write been integrated in the base line of the code?
    - If not, is it because none have been proposes? (or is it because the community does not allow people to become contributors?)
- Are the justifications for accepting or rejecting correction propositions explicit?
- Are the version control system, the packaged distributions, and patch files organized appropriately in order to retrieve others contribution in an iterative, selective or holistic fashion?

Remark to consider when creating Indicators:

- See remarks for indicators for Product Manager viewpoint of Capability of requirement and change management

Architect, Analyst, Developer and Tester:

Viewpoint: Architects, analysts and developers are interested to know about
- The low level procedures to follow when submitting a proposed correction (e.g how to generate the patch file, what test suite and test report should be included, ...)
- The change management tools used and whether they are configured to automate certain steps of the corrections proposal/acceptance process.
- The test information to provide when proposing corrections (success/failure of test runs, coverage info)

Measurement Questions:
- Is there a low level procedure that explicitly specify
    - The testing requirements to satisfy for a correction to be accepted?
    - How to generate a patch files for submitting a correction?
    - What to provide with the patch file for submitting a correction (eg. test report)? Is there a script or a standard command line for easily generating a proper patch and even a proper correction proposal?
- Is there a test script to run the required regression test suite?
- Is the format of new tests to add to a regression test suite explicit?

Remark to consider when creating Indicators:
- See remarks for indicators for Product Manager viewpoint of Capability of requirement and change management

Technical Writer:

Viewpoint: Architects, analysts and developers are interested to know about
- Documentation requirement when submitting the implementation of an enhancement (in particular, if that enhancement adds functionality to the F/OSS component)
- Template to use for writing documentation

Measurement Questions:
- Are there documentation requirements that explicitly specify the type of documentation to submit with the implementation of an enhancement?
- Do templates exist for writing documentation in the expected formats?

Remark to consider when creating Indicators:
- See remarks for indicators for Product Manager viewpoint of Capability of requirement and change management

11.Software Processes – Capability of Release Management

Definition: The degree of capability with which an F/OSS endeavor handles its release management activities.

- NOTE: An F/OSS endeavor is considered more capable to manage releases if it explicitly proposes procedures for initiating, planning, and packaging new releases of its F/OSS component.

Assessment goals for Indicators of Capability of Release Management:

- Evaluate the degree of risk related to the capability of an F/OSS endeavor to handle release management process based on explicit procedures and past evidences of release management activities.

Interest and Questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- The person who has the overall responsibility of supervising new major releases
- The predictability of future major releases (either in term of absolute time or in relation to a roadmap)
- The consistency in respecting the proposed planning of major releases
- The procedure to assign the priority to enhancements and corrections to include in the future major release
- The procedure to plan the implementation of selected enhancements and corrections (eg roadmap with milestones or or other planning doc?)
- The platform/mechanism used to discuss backward compatibility issues with previous, supported major releases
- The predictability of the quality of a F/OSS component evolving through the release management cycle of a initial major release. (e.g. is there a standard process to change the label of a release from alpha to beta to RC1 to RC2 ... to stable)

Measurement Questions:
- Is a person assigned to the role of (major) release management?
- Are the responsibility of the (major) release manager explicit?
- Are new major versions of F/OSS component released in a predictable fashion? (either on an absolute time scale or in relation to an explicit roadmap or in terms of number or severity of corrections)
- Have previous releases respected the planning? (within acceptable delays)
- Is there a platform to discuss and to debate the enhancements and corrections to include in a future major release and also to debate about backward compatibility issues with previous, supported major releases?
- Is there a platform to discuss and to debate the plan of a future major release?
- Is the quality of a F/OSS component predictable from its release label for initial major releases? (e.g. where x < y means x has less bugs than y then for instance, the label could be apha < beta < RC1 < ... < RCn < stable) (it is possible to study the trend of the number of bug reported vs time for each label and observe if there is a trend over several major releases)

Remark to consider when creating Indicators:
- See remarks for indicators for Product Manager viewpoint of Capability of requirement and change management
- (major) and below (minor) are between parentheses to show that it is also acceptable to have just a single release manager that handles both minor and major release management

Project Manager:

Viewpoint: The product manager is interested to know about
- The person who has the overall responsibility of supervising new minor releases

- The predictability of future minor releases (either in term of absolute time or in relation to a roadmap)
- The consistency in respecting the proposed planning of minor releases
- The procedure followed to decide what patches/corrections to include in a future minor release.
- The platform/mechanism used to discuss and to decide whether new functionality may be implemented in a minor release.
- The predictability of the quality of a F/OSS component evolving through the minor release management cycle. (e.g. are the # of bugs decreasing between each minor releases, for instance, with a logarithmic trend)
- The procedure for packaging a new release (e.g. does it force the inclusion of other work products related to a F/OSS component, in particular, documentation, tests, change logs, etc.?)

Measurement Questions:

- Is a person assigned to the role of (minor) release management?
- Are the responsibility of the (minor) release manager explicit?
- Are new minor versions of F/OSS component released in a predictable fashion? (either on an absolute time scale or in relation to an explicit roadmap or in terms of number or severity of corrections)
- Have previous releases respected the planning? (within acceptable delays)
- Is there a procedure to select patches to include in a future minor release?
- Is there a platform/mechanism to discuss and to debate about the implementation of new functionality in a future minor release?
- Is the quality trend of a F/OSS component predictable between minor releases? (e.g. study the evolution of bug numbers vs time between minor releases)
- Is there an explicit procedure that describes what work products to include in related the packaged distribution of a new release?
- Do new releases include work products other than the source code of F/OSS component, for instance, binaries, installers, documentation, tests, change logs, etc.

Remark to consider when creating Indicators:

- See remarks for indicators for Product Manager viewpoint of Capability of requirement and change management

Architect, Analyst and Developer:

Viewpoint: Architects, analysts and developers are interested to know about

- The low level procedures to follow to patch several releases where the same fault is present?

Measurement Questions:

- Is there an procedure generally followed for patching several minor/major releases with the same problem?

Remark to consider when creating Indicators:

- See remarks for indicators for Product Manager viewpoint of Capability of requirement and change management

12. Software Processes – Capability of Support and Community Management

Definition: The degree of capability with which a F/OSS endeavor handles support activities to its F/OSS community appropriately and efficiently.

- NOTE: The F/OSS community includes all people who have contributed to the F/OSS endeavor including users who post questions on support platforms.

Assessment goals for Indicators of Capability of Support and Community Management:

- Evaluate the degree of risk related to the capability of an F/OSS endeavor to handle its support and community management process based on explicit procedure and pas evidences of support and .

Interest and Questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- Events where the F/OSS community members meet
- Availability of commercial services for upgrade to new major releases of the F/OSS component
    - The governance rules that regulates the F/OSS community (e.g. meritocracy to earn commit right, democratic election to access certain leadership positions, ...)
- Consistency of actions promoting and marketing the F/OSS component (e.g organizations pushing the F/OSS component, publication of press release in F/OSS news when new releases are available, ...)
- Financial support from enterprises and individual donors
- List of reference users of the F/OSS component, possibly enterprises, and the mission criticality of the F/OSS component for these users.
- Involvement of researchers in the implementation of new features in the F/OSS component

Measurement Questions:
- Are they events where F/OSS community members can meet to debate on various issues and to present their current work in the F/OSS endeavor?
    - Are the event well attended?
    - Are some events connected to Industry?
    - Are some events addressing technical aspect of the F/OSS component?
- Are commercial services available for upgrading between major releases of the F/OSS component?
    - Are these services provided by committers?
    - Are these services worldwide (or covering a desired geographical location and in the desired language)?
- Are the governance rules explicit?
    - Do they allow for an open debate within the community
    - Are positions accessible through elections?
- Are enterprises involve in the promotion and marketing of the F/OSS endeavor?
- Are events connected to the F/OSS endeavor published in technical and general FlOSS news?
    - Are new releases announced on general F/OSS news site such as Slashdot?
- Is the F/OSS endeavor supported financially (beside employees donating hours of work)
    - Are enterprises financially supporting the F/OSS endeavor?
    - Are individuals financially supporting the F/OSS endeavor?
- Is the F/OSS component already used in other enterprises' software products?
    - Are these software product important to the mission on these enterprises?
- Are researchers implementing their research results in the F/OSS component?

Remark to consider when creating Indicators:
-

Project Manager:

Viewpoint: The product manager is interested to know about

- Availability of commercial services that provide expert advice on the internals of the F/OSS component and more specifically, on the particular version of interest of the F/OSS component.
- Guaranteed rapidity of commercial services on the F/OSS component

Measurement Questions:

- Are commercial services available on the internals of the F/OSS component, in particular, the version of interest?
    - Are these services provided by committers?
    - Are these services worldwide (or covering a desired geographical location and in the desired language)?
    - Do these service guarantee rapidity in treating problems?

Remark to consider when creating Indicators:

-

Architect, Analyst and Developer:

Viewpoint: Architects, analysts and developers are interested to know about

- Help or assistance to integrate the code contributor community
- Availability of a knowledge base on technical aspect of the version of interest of the F/OSS component (including changes from previous major release and changes from previous minor releases) (eg technical documentation + searchable forums, website that collects tutorial, archive of technical IRC sessions, ...)
- Availability of a platform/mechanism to ask free technical support questions whose answer are currently not found in knowledge base (e.g mailing list, web forum, etc.)
- Average time in which support questions have been answered on freely available platform

Measurement Questions:

- Is the current community helpful with, or does it provide assistance for, new comers who would like to become contributors? (e.g remote coaching, etc.)
- Are there various repository of information for asking technical questions, in particular, dedicated to developers of the F/OSS component?
- Is there a free mechanism/platform to ask technical questions?
    - Through mailing list dedicated to developers?
    - Via a web forum?
    - Via an issue tracking system?
- What is the percentage of questions that find answers?
- What is the average time in which questions find answers (for those that are answered)?

Remark to consider when creating Indicators:

-

13.Tools – Compile-time and Runtime Dependencies – FlOSS Endeavor Compatibility

Definition: the degree to which a FlOSS endeavor is compatible with F/OSS endeavors on which it depends

- NOTE: Compatibility also includes the notion of replaceability that is, whether or not the FlOSS endeavor could easily substitute its dependence on F/OSS components produced by other F/OSS endeavors.

Assessment goals for Indicators of FlOSS endeavor Compatibility (with dependencies):

- Evaluate the degree of risk related to the compatibility between the F/OSS endeavor of interest with other F/OSS endeavors on which it depends, usually because the F/OSS component of interest depends on F/OSS components produced by these other F/OSS endeavors.

Interest and Questions for the various viewpoints

Product Manager:

Viewpoint: The product manager is interested to know about
- Coherence between the licenses of dependencies and the on of the FlOSS component of interest
- Closeness between dependencies and the FlOSS endeavor of interest (eg belong to the same F/OSS project or same F/OSS foundation)
- Members working on the F/OSS component of interest and also on dependencies
- The synchronization between releases of the F/OSS component of interest and those of its dependencies
- The replacement option (or the lock-in with another F/OSS endeavor)

Measurement Questions:
- Are licenses of dependencies compatible with that of the F/OSS component of interest. (i.e., a dependency's license must be equally or less permissive)
- Are dependencies under the same F/OSS project or F/OSS foundation than the F/OSS endeavor of interest?
- Does community of the F/OSS endeavors of interest share members with its F/OSS dependencies?
- Are the major releases of the F/OSS component of interest synchronized with those of its dependencies (with a small delay)
- Could the F/OSS endeavor of interest substitute its dependencies on other F/OSS endeavors, in particular, could it the F/OSS component of interest substitute its dependence on other F/OSS subcomponents?
    - What are the F/OSS endeavors on which the FlOSS component of interest is technologically dependent? (e.g. Plone is technologically dependent on Zope and Python or Eclipse plugins are technologically dependent on Eclipse)
    - What are the F/OSS endeavors on which the F/OSS component of interest is functionally dependent?

Remark to consider when creating Indicators:
-

Architect, Analyst and Developer:

Viewpoint: Architects, analysts and developers are interested to know about
- The coupling from the F/OSS component of interest and its dependencies

Measurement Questions:
- Are the interactions with dependencies limited to a few source files in the code of the F/OSS component of interest?
- Are interactions with dependencies limited to a few library calls?

Remark to consider when creating Indicators:

•

APPENDIX B: SAMPLE TEMPLATE FOR DESCRIBING THE ASSESSMENT OF AN FLOSS ENDEAVOR

APPENDIX C: TEMPLATE FOR RECORDING IMPORTANT LOW-LEVEL OR UNANTICIPATED ACTIONS

This template allows the various people involved in an assessment of an F/OSS endeavor to record actions they or others have performed. The actions to record are either the important non-trivial actions (or decisions) or the unanticipated action that were not planned in the original workflow.

Filling diligently the table template below will help to fulfill the traceability requirement.

F/OSS Endeavor Name:			
Person:		Date:	
Assessment Method Name & version:	*Assessment for Full F/OSS Collaboration – Version 1*		
Task Name:		Workflow Operation #:	
Description of Actions:			
Action 1 description			
Action 2 description			
...			

Appendix D: Template for Specifying Indicators

Quality Attribute

<Quality attribute this indicator applies to. Use the name from D1.3>

Contact person

<Name and institution of the person responsible for this indicator definition>

Quality level definition

This section defines the meaning of the four quality levels, Black, Red, Yellow, and Green in the context of this particular indicator. They should be specific to the particular quality attribute.

Black

<Definition of the Black level>

Red

<Definition of the Red level>

Yellow

<Definition of the Yellow level>

Green

<Definition of the Green level>

Metrics

This section is used to refer to metrics relevant to the indicator.

Metrics from D1.3

List here the metrics defined in D1.3 that are necessary to calculate the indicator. Please use the exact names from D1.3.

- *<MetricOne>*
- *<MetricTwo>*

New required metrics

Describe here any metrics that you consider necessary in order to evaluate the indicator, but that are not defined in D1.3. Use a description list as follows:

<MetricThree>

<Explanation of the third metric.>

<MetricFour>

<Explanation of the fourth metric.>

Unused metrics from D1.3

List here the basic metrics that were defined in D1.3 for the quality attribute, but that you decided not to use. Add a brief explanation telling why they are unsuitable. Use a description list as follows:

<UnusedMetricOne>

<Why UnusedMetricOne was not used>

<UnusedMetricTwo>

<Why UnusedMetricTwo was not used>

Indicator evaluation

Describe here the rule or formula used to evaluate the indicator. The inputs are the metrics listed in the previous section. The output is a value from the set {Black, Red, Green, Blue}.

Changes to QualOSS model

Describe the changes to the QualOSS model you did during measurement or indicator definition, or changes that you think would be necessary in the future.

Issues for advanced metrics

Discuss any issues you may have observed regarding the introduction of advanced metrics in later phases of QualOSS. This includes problems that may arise, or opportunities we may have.

Additional Comments / Problems

Add here any additional comments you may have. This includes but is not limited to problems that may occur while evaluating the indicator, and issues related to data availability and reliability.