a STReP Project (Specific Targeted Research Program) #033547
Sponsored through Framework Programme Sixth (Call 5) by

Document Information

Version: 1.0
Date : Jan 12, 10
Pages : 33

Author(s):
Jean-Christophe DEPREZ (CETIC)

Project Dates:
Start: Sept. 2006
End: Nov. 2009
Duration: 39 months

Coordinating Organization:
CETIC

Coordinating Person:
Jean-Christophe DEPREZ

The QUALOSS Consortium consists of: CETIC (BE), FUNDP (BE), URJC (ES), Fraunhofer IESE (DE), ZEA Partners (BE), MERIT (NL), AdaCore (FR), PEPITe (BE)

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

Title:

QualOSS Final Activity Report

Disclaimer:
The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability."

Copyright 2008, Jean-Christophe DEPREZ (CETIC)

TABLE OF CONTENTS

1. Project Execution 3
1.1 Project Objectives 3
1.2 The QualOSS Consortium 4
1.3 Work Performed And Results 5
1.3.1 Goal-Question-Metric and the QualOSS Assessment Methodology 5
1.3.2 Assessment Process of the QualOSS Assessment Methodology 9
1.3.3 The Standard QualOSS Assessment Method and Assessment Results 11
1.4 Comparison to Related Works 22

2. Dissemination and Use 25
2.1.1 Case Studies 25
2.1.2 Exploitation scenarios for consultant 29

3. References 32

1. Project Execution

Section 1.1 presents the overall project objectives of QualOSS. Section 1.2 reviews the partners involved. Section 1.3 summarizes the work performed to reach the objectives and more importantly, presents an overview of the end results. Section 1.4 compares QualOSS results to similar work.

1.1 Project Objectives

The strategic objective of this project is to enhance the competitive position of the European software industry by better leveraging on free/open source software (F/OSS). To achieve this strategic goal, QualOSS proposes to build a methodology to benchmark the quality of open source software in order to assist companies in their strategic decision to integrate F/OSS.

Thus, the two main objectives of QualOSS are :

1. To build an F/OSS assessment methodology based on the Goal-Question-Metric paradigm (GQM)
2. To develop associated tools to automate the tedious operations of an assessment

Initially, QualOSS recognizes that solely assessing the quality of the product or code of a F/OSS would not provide the complete and adequate picture desired by enterprises. Indeed, it also considers important to assess the quality of the community around the F/OSS, as it may bare on the worthiness to integrate F/OSS. QualOSS also observes that the evolvability and robustness of F/OSS projects directly impact the worthiness of integrating F/OSS for enterprises. In consequence, the assessment methodology should stress its focus on these two quality attributes.

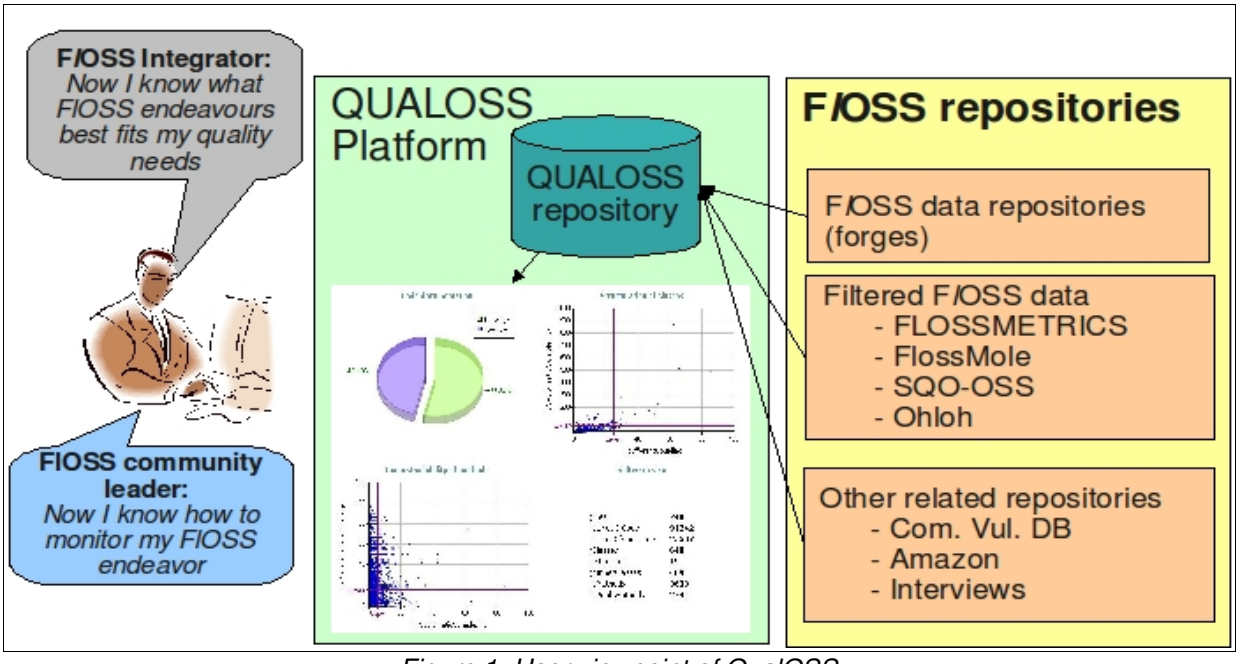

Figure 1: User viewpoint of QualOSS

Figure 1 illustrate QualOSS's original objective. It shows how the QualOSS platform (tool part) will help enterprises in their decision about F/OSS. Alternatively, it will allow F/OSS developers to browse QualOSS assessment results in order to understand how enterprises assess the robustness and evolvability of their F/OSS project. The arrows from the F/OSS repositories (yellow box) to the QualOSS repository (green

cylinder) represent the flow of a QualOSS assessment where analysis and measurement tools are applied to F/OSS data to retrieve and to summarize important information about the robustness and evolvability of a F/OSS project.

As detailed in sections 1.3 *Work Performed And Results* and 1.4 *Comparison to Related Works*, QualOSS is different from current assessment techniques in the fact that its methodology recommends assessing various aspects of a F/OSS development endeavor, in particular, product/component including the source code, community, software process, and the tools and dependencies. Furthermore, to perform broad assessments, QualOSS plans to take advantage of information widely available in various F/OSS repositories such as the data produced by F/OSS projects but also in other repositories such as Common Vulnerability and Exposure data (CVE) in the national vulnerability database or publication databases.

Although QualOSS plans to automate the tedious procedures of an assessment, it is unlikely that every aspect can be computed automatically. User guides will specify how to perform the manual parts of an assessment. Furthermore, the visualization tool will present information so results of an assessment can easily be interpreted.

1.2 The QualOSS Consortium

The QUALOSS consortium consists of 8 partners from 5 member states,namely, Belgium, Germany, France, Spain, and The Netherlands.

Part. #	Name	Country	Expertise
1	Centre d'Excellence en Technologies de l'Information et de la Communication (CETIC) Project Coodrdinator	Belgium	Measurement, Qualitative Modelling, Empirical Studies, Open Source
2	Facultés Universtaires Notre Dame de la Paix - FUNDP (FUNDP)	Belgium	Measurement, Qualitative Modelling, Empirical Studies, Process Assessment
3	University of Rey Juan Carlos (URJC)	Spain	Qualitative and Quantitative Analysis of F/OSS projects focus on the automated data collection
4	Franhofer IESE (IESE)	Germany	Measurement, Qualitative Modelling, Empirical Studies, Process Management
5	ZEA Partners (ZEA)	Belgium	Open Source Use in Industry Context, ZOPE & Python technologies
6	MERIT	Netherlands	Analysis of F/OSS focused on the economic impact.
7	ADACORE	France	Open Source Development, Use of F/OSS components
8	PEPITe	Belgium	Data mining

1.3 Work Performed And Results

The first objectives of QualOSS is to develop a F/OSS assessment methodology based on the Goal-Question-Metric paradigm (GQM). Consequently, it is important to understand GQM and the spirit in which it has been applied. Section 1.3.1 gives a brief review of GQM and then shows how the QualOSS methodology proposes to apply it. Then Section 1.3.2 summarizes the assessment process prescribed by the QualOSS methodology. Finally, Section 1.3.3 briefly describes the Standard QualOSS Assessment Method, which rigorously follows the QualOSS methodology, and then presents samples of assessment results using the Standard QualOSS Assessment Method.

1.3.1 Goal-Question-Metric and the QualOSS Assessment Methodology

The Goal-Question-Metric paradigm (GQM) was initiated by Basili (Basili, 1992), to help enterprises to measure and control development project efficiently. Until then, in many cases, companies tended to collect many useless measures and then wasted significant effort trying to use these measures to control various aspects of their software project. Converse to this approach, GQM proposes to only collect measures for which explicit measurement goals are stated. Measurement goals derive directly from business needs. For example, a business need may be that management wants to increase customer satisfaction. This business goal can then be transformed into a measurement goal, for example, noting that user satisfaction can partly be measured by the number of calls from customers to the call center. Consequently, in this instance, a measurement goals would be the change in the rate of customer calls to the call center. Clearly, other measurement goals that indirectly influence the rate of customer calls such as number of bugs or internal measures of software quality could also become explicit measurement goals.

Once measurement goals are known, GQM asks to elaborate a set of questions whose answers clearly ask for the needed measures. In the example above, some questions would be *what is the number of customer calls per month for each older version of the product?* and *what is the the number of customer calls per month for the new version of the product?* and *what is the trend/change in the number of calls over the various versions in chronological order? (stable, increasing, deceasing)*. From these questions, it then becomes easy to identify the measures (or metrics) to collect at the various stages of the life cycle of a software product.

To help specify measurement goals adequately, GQM provides a measurement goal template that requests the following pieces of information.

Issue	Short description of the problem that requires an assessment
Purpose	The purpose of the analysis
Context	The context (and thus, planned area of validity) of an analysis
Object	The object to assess
Viewpoint	The viewpoint or role from which the analysis is done
Quality focus	The object's quality attribute covered by the analysis

Issue: In the QualOSS Methodology, the *issue* is to help enterprise faced with difficulties in selecting the most appropriate F/OSS endeavor from which to acquire a F/OSS component.

Purpose: The *purpose* is to evaluate the risk to integrate a F/OSS component.

Context: Regarding the *context*, it is important to understand that GQM was initially built to be applied in very specific contexts, for instance, on a particular software product of a company. In such a specific context, it is quite simple to clearly obtain clear business and measurement goals from the company. Subsequently, the questions and measures are also very explicit and unambiguous.

To specify a *context* that should be specific enough to apply GQM appropriately, the QualOSS methodology recognizes that the various F/OSS integration contexts may require different kind of assessments. For example, one who wants to install and use a F/OSS application without participating to the corresponding F/OSS project likely needs a different type of assessment than one who plans to integrate a F/OSS component in an large software application and who plans to actively participate to the F/OSS project. Consequently, the QualOSS Methodology proposes to partition the various F/OSS integration along different dimensions. Each subspaces created by these partitions is much more likely to embody a similar kind of business needs and measurement goals. The different dimensions of a F/OSS integration situation are:

- Integration Purpose: Integration of a F/OSS in a software product, in a service or in an infrastructure
- Collaboration Context: F/OSS full collaboration, F/OSS takeover, F/OSS fork, F/OSS exploit
- Assessment Mode: product comparison, version comparison

The dimension *Integration Purpose* suggests that an assessment may differ if an enterprise wants to integrate a F/OSS component in one of its software product, in one of its services, or on its infrastructure.

The dimension *Collaboration Context* suggests that an assessment may differ if an enterprise wants to establish a long term collaboration with an existing F/OSS community (F/OSS full collaboration), or wants to to take over a small or dead F/OSS community (F/OSS take over), or wants to fork the code based without collaborating with the F/OSS community in the future (F/OSS fork), or wants to merely install and use a F/OSS component without interest in its source code nor in collaborating with the existing community (beside eventually reporting a bug) (F/OSS exploit).

The dimension *Assessment Mode* suggests that an assessment may differ if one assesses two (or more) versions of the same F/OSS component (version comparison) or compares two complete different F/OSS component. In a version comparison assessment, a F/OSS component is already selected and the object of the assessment is to help select the most appropriate version. Such a situation takes place when one wants to know whether to upgrade from an older to a newer version of a F/OSS component. In the product comparison, two completely different F/OSS components are compared.

For a F/OSS integration context narrow enough for an adequate application of GQM, the QualOSS Methodology requires to select a particular subspace where a single value for each dimension is chosen and then a set of measurement goals are derived for that narrower context. For example, subsection 1.3.3 explains that the Standard QualOSS Assessment Method applies the GQM for a F/OSS integration context defined as { Integration in a software product, F/OSS full collaboration, product comparison }

Object: The next item in the GQM template is the *object* to assess. Based on interaction with enterprises, it is clear that for certain F/OSS integration situations, more than product quality is needed. Thus, to formalize the *object* to assess on sound basis, the QualOSS Methodology observed that F/OSS development are human activity. Activity Theory, which has been elaborated an used for more than a century, provides an adequate framework describing the various element involved in an human activity. It therefore provides the needed formalism to define the *object* of an assessment, namely, a F/OSS endeavor.

An ***F/OSS endeavor*** is the undertaking of community members using tools and following software processes to produce work products related to one or more F/OSS components. (Deprez et al., 2007)

*NOTE 1: In this definition, a **F/OSS component** is defined by a set of source code files.*

*NOTE 2: In this definition, the notion of **tools** is to be taken broadly. The set of tools includes process support tools such as version control or bug tracking systems and it also includes software libraries used by a F/OSS component, which may be produced by another F/OSS endeavor.*

NOTE 3: Henderson-Seller also defined the concept of Endeavor that has similarities with the definition above, although his definitions was not limited to F/OSS. It is worth noting that our definition was developed completely independently from Henderson-Sellers' one. (Henderson-Sellers, 2002)

Formally, we define an F/OSS endeavor as a tuple of 4 sets:

- *Community Members* is a set of community members who contribute to the F/OSS endeavor,
- *Work Products* is a set of work products produced by community members, including the source code of the F/OSS components
- *Software Processes* is a set of software processes pertinent to the F/OSS world followed by community members to produce work products, and
- *Tools and Libraries* is a set of tools such as libraries (dependencies) 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

When performing an assessment, it is critical to scope a F/OSS endeavor appropriately. In other words, the four set defined above should make sense. For example, it would be incorrect to measure community based on the complete set of members who participate in a F/OSS project where only a small sub-component of the F/OSS component would be measured. In such a scenario, only the set of community member who contributed to the small sub-component should be considered for the community measurement.

The last two items of the template are viewpoint(s) and quality focus.

Viewpoint: Viewpoints refer to the various roles of people in an enterprise involved in a F/OSS integration, traditional roles are: product manager, project manager, analyst, developer, tester, documentation/technical writer, system administrator.

Quality focus: The quality focus can vary depending on the F/OSS integration context and on the specific needs of an enterprise. In Section 1.3.3, two important quality aspects of a F/OSS endeavor will be the focuses named robustness and evolvability. However, the QualOSS Methodology can also be followed for other quality focuses of a F/OSS endeavor.

In summary template to follow the QualOSS Methodology is therefore as follows.

Issue	Difficulty in selecting the most appropriate F/OSS endeavor from which to acquire a F/OSS component
Purpose	Evaluate the risk to collaborate to integrate a F/OSS component in the given context, for the given viewpoint and quality focus
Context	The context must be defined from the 3 dimensions of a F/OSS acquisition: • *Usage = { integrate in an infrastructure, integrate in a product, integrate in a service}* • *Mode = { product comparison, version comparison }* • Collaboration = { F/OSS exploit, F/OSS fork, F/OSS takeover, F/OSS full collaboration}
Viewpoint	The viewpoint or role encountered in an enterprise project where a F/OSS integration is needed, • *Product Manager* (with a long term Management viewpoint), • *Project Manager* (with a short term management viewpoint), • *Product Architect* (with a technical long term viewpoint), • *Developer, Analyst, Tester, Technical Writer* (each with a technical short term viewpoint) • System Administrator (with a short term technical viewpoint)
Object	*A F/OSS endeavor (appropriately scoped)*
Quality focus	*Quality attribute of a F/OSS endeavor such as robustness, evolvability, maturity, dependability, productivity, etc.*

Once the template has been instantiated to a particular F/OSS integration context, and the viewpoint and quality focus for that context have been elicited, it is then possible to specify the high-level measurement goals.

The next step is to refine each top level measurement goal into a set of low-level measurements sub-goals for each element of a F/OSS endeavor, namely, product, community, software process, and tools and dependencies as illustrated in Figure 2. An example of such a refinement is presented in Section 1.3.3 for the robustness and evolvability of a F/OSS endeavor.

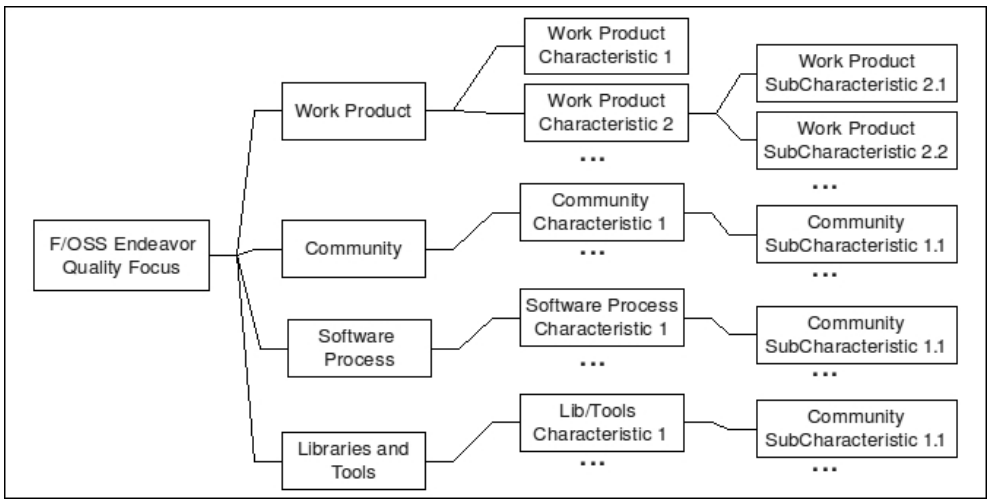

Figure 2: Usual QualoSS quality model definition.

The low-level measurement sub-goals is then formed from a leaf of the tree, the given F/OSS integration context and a selected viewpoint (role). Furthermore, as specified in the GQM template of the QualOSS Methodology above, the purpose of each measurement sub-goal is to evaluate the risk of integrating a F/OSS component. To evaluate this risk, a set of questions of interest are elicited for each low-level measurement goal.

The QualOSS Methodology thus expects that answers to questions will help assess risk of integrating a F/OSS component. Consequently, the QualOSS Methodology suggests that answers to question use a 4-value scale that can be visualized using colors green, yellow, red and black where green indicates no risk (or negligible risk), yellow (small risk), red (medium risk) and black (high risk). To provide answers to questions, measures are aggregated and then projected into the 4-value scale. This aggregation and projection defines what is referred to as an *indicator*. Clearly, some indicators may be more important than others for assessing risk thus each indicator can be assigned a weight.

It is also worth mentioning that question are formulated so their answers can be measured. Given the accurate low-level goals based on a narrow integration context, a single viewpoint and single quality focus, the measures for answering it can be precisely defined. In consequence, tools (automated, manual or partly automated) can be developed to take the measures.

Beside measure values, it is also useful to characterize measurement with a *status* to specify if it was performed as anticipated or if special circumstances were encountered, for example, the measured dataset was incomplete, noisy, inexistent. Alternatively, the status can also be used to show that a measurement tool malfunctioned. The QualOSS Methodology recommends to use positive value for status where measure values can reliably be used to compute indicators and negative status value for cases where the measure value is quite unreliable and should not be considered by an indicators.

Similarly, the indicator value obtained from a set of measure values *M* should be accompanied by a measure coverage. This coverage will depend on the measurement status for the measures in *M*.

An indicator value is then further characterized by a coverage and the weight of importance of the indicator for answering a question. The QualOSS Methodology proposes to combine these two properties to obtain the *confidence* factor to attribute to an indicator value. This *confidence* factor specifies how much importance an indicator value should be given when using it to assess the risk (and answering the question).

After having gathered all these pieces, one has applied GQM as requested by the QualOSS Methodology. In addition to the application of GQM, a QualOSS Assessment Method must also follow a particular assessment process described in the next subsection.

1.3.2 Assessment Process of the QualOSS Assessment Methodology

This section presents 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.

As illustrated in Figure 3, the first 4 tasks are performed in sequence while the last task of supervision is transverse thus it is executed throughout the other tasks of the assessment process. The arrows show the input/output dependencies between tasks.

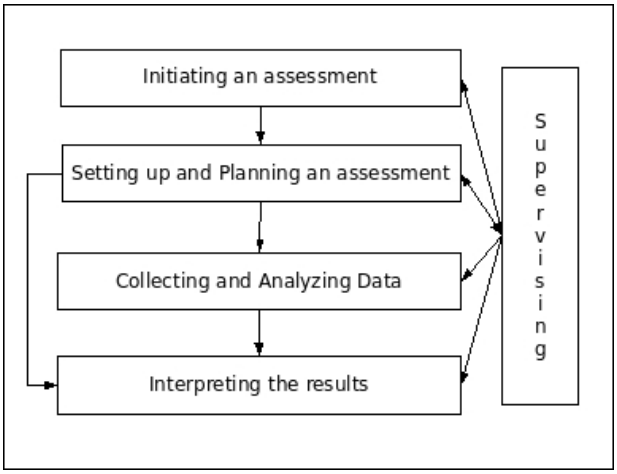

Figure 3: Dependencies between tasks of the assessment process.

Several facts are worth noting

- There exists a loose correspondence between these 5 tasks and the 6 activities of the evaluation reference model from the up coming SQuaRE 25000 series (replacement of ISO9126). The main difference is that SQuaRE split the requirement elicitation and the specification of an assessment where as these two activities are performed jointly in the setting up and planing phase in the QualOSS assessment process. Other important difference is that the QualOSS assessment process believes that a supervision take for supervising the assessment should be included to verify that all required recording and actions take place during an assessment. On the other hand, the ISO assessment process does not explicitly include supervision as part of the process.
- The current version of the assessment does not include a feedback loop. In practice however, the accurate scope of the F/OSS endeavor is hard to identify during the setting up and planning task. In consequence, it is possible that new datasets are identified only in subsequent tasks. In such a case, it is important that the evaluator(s) determine the impact of the change in scope. If needed, certain actions of the previous tasks may need to be redone in order to present coherent results.
- To increase the soundness of the QualOSS assessment process, task description uses the activity theory framework (Bjorke, 2005). In particular, task description are related to the various element of the Activity Theory framework illustrated in Figure 4. Activity Theory has been elaborated for more than a century and is used extensively in sociology, education to study and understand human governed activities. As shown in (de Souza, 2003), it has also been used in the Software Industry to study the behavior of developers. Although a QualOSS assessment can be partly automated, they remain a human-driven activity. Assessors are always needed to specify the input, guide an assessment and verify the results hence the Activity Theory framework applies quite well to this context.

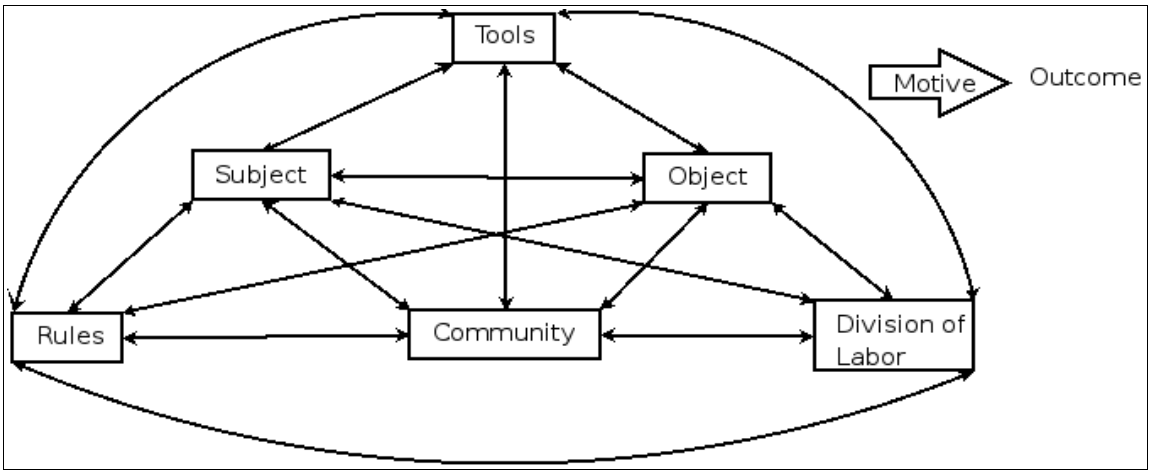

Figure 4 - An activity system: its elements and their interactions when an activity takes place.

Information about objectives and outcomes expected for each of the 5 tasks in Figure 3 are detailed in deliverable D4.5 of the QualOSS project. Furthermore, D4.5 highlights the connection with the part of the activity system framework covered. Finally, D4.5 also mentions the procedures, tools, and techniques expected to help in conduct each task.

1.3.3 The Standard QualOSS Assessment Method and Assessment Results

1.3.3.1 Following the QualOSS Methodology to Build the Standard QualOSS Assessment Method

From its inception, the QualOSS project believes that many enterprises will integrate F/OSS code in their application. This belief is supported by a Gartner report that predicts that 80% of all commercial software will include elements of F/OSS technologies (Gartner, 2008). In the majority of cases where F/OSS code is integrated in software application, the enterprise will invariably need to collaborate closely with the F/OSS endeavor that produce the F/OSS code of interest. In other words, in the future, the following integration context will get increasing attention:

{ Integration in a software product, F/OSS full collaboration, product comparison }

In this integration context, important quality aspects relate to the robustness and the evolvability of a F/OSS endeavor, where robustness means the capability to solve current problems and evolvability, the capability to solve future problems and issues likely to arise. In brief, an enterprise will only be interested to collaborate hence use code from F/OSS endeavor that are capable of solving its past and current problems (robustness) and its potential problems in the future (evolvability).

Furthermore, in this integration context, the most prominent viewpoints are the following roles:

- Product manager who has a long term management view on the enterprises software product hence who wants to only include F/OSS code with a F/OSS endeavor that has a great chance to thrive in the future,
- Project manager who has a short term management view on the initial project that will integrate a F/OSS component in the enterprise's software product.
- Architect who has a long term technical view on the fact that the target enterprise software product will be able to integrate the selected F/OSS component in the current version and that this will remain true with future version of the F/OSS component.

- Analyst, Developer, Tester who have a short term technical view on the integration of the current F/OSS component in the enterprise software product.
- Technical Writer who has a short term non-technical view on the appropriateness of the F/OSS component documentation to leverage on for writing the documentation of the enterprise software product.

Consequently, the Standard QualOSS Assessment Method was built to address the integration context, the quality focus and the viewpoints mentioned above. In the GQM template proposed by the QualOSS Methodology, this give the following:

Issue	Difficulty in selecting the most appropriate F/OSS endeavor from which to acquire a F/OSS component
Purpose	Evaluate the risk to collaborate to integrate a F/OSS component in the given context, for the given viewpoint and quality focus
Context	The context must be defined from the 3 dimensions of a F/OSS acquisition: • Usage = *integrate in a software product* • Mode = *product comparison* • Collaboration = *F/OSS full collaboration*
Viewpoint	The viewpoint or role encountered in an enterprise project where a F/OSS integration is needed, • *Product Manager* (with a long term Management viewpoint), • *Project Manager* (with a short term management viewpoint), • *Product Architect* (with a technical long term viewpoint), • *Developer, Analyst, Tester, Technical Writer* (each with a technical short term viewpoint) • System Administrator (with a short term technical viewpoint)
Object	*A F/OSS endeavor (appropriately scoped)*
Quality focus	*Robustness and Evolvability of a F/OSS endeavor* • *Robustness means the capability that the F/OSS endeavor displayed in solving past and current problems* • *Evolvability means the capability that the F/OSS endeavor will likely display in solving future problems*

To continue applying the requirement of the QualOSS Methodology, the high level quality forcus “robustness and evolvability of a F/OSS endeavor” must be refined for each of the 4 elements of a F/OSS endeavor, that is, product, community, software process, and tools and dependencies. This refinement is illustrated in Figure 5.

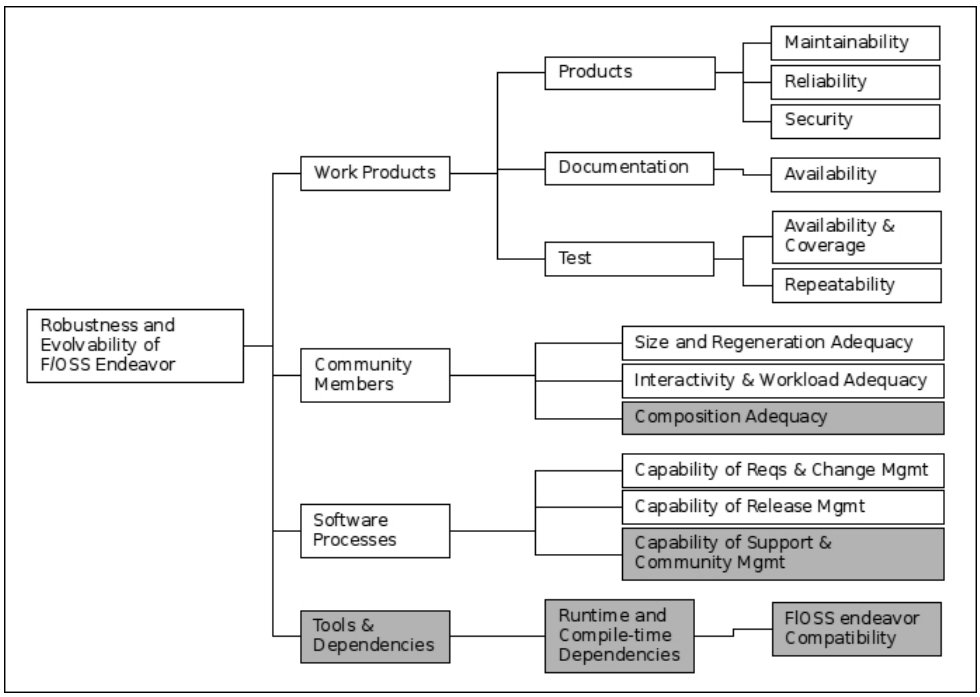

Figure 5: Hierarchical Quality Model of QualOSS Final Activity Report used to assess risks related to the robustness and evolvability of an FlOSS endeavor.

The gray nodes indicate that due to time consideration, they could not yet be included in the latest version (v1.1) of the Standard QualOSS Assessment Method provided in deliverable D4.5.

At this point, it is possible to define all the low level measurement sub-goals by selecting a leaf in the quality model tree above, an viewpoint. Furthermore, all sub-goals target the evaluation of risk of integrating a F/OSS component for the given integration context, given a selected viewpoint and for a quality focus found in the leaf of the quality model tree.

For each sub-goal, a set of questions whose combined answer assess risk of integration are elicited. Below is the list of questions related to the

- the integration context { Integration in a software product, F/OSS full collaboration, product comparison },
- the quality focus Maintainability, and
- the viewpoint of the Product Manager

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)?

Due to space consideration, questions for other viewpoints and other quality focus are not presented in this document, they are found in deliverable D4.5 of the QualOSS Project or can also be viewed by browsing an assessment at http://ingrid.cetic.be:33323/qualoss_assessment/.

The next step in following the QualOSS Methodology to build the Standard QualOSS Assessment Method is to create a set of risk indicator to answer each question in a risk-oriented way. Although it is possible to answer a question using several risk indicators, it is possible that only a single indicator provides enough information to answer the questions hence the set of indicators is a singleton with inly 1 indicator in the set.

Again, due to space consideration, only one indicator is presented, in particular, the one to answer question WP-Ma-1 in a risk-oriented way.

Indicator

Name: *Percentage of accepted enhancement proposals*

Description:

- Green: more than 10% of the enhancement proposals are accepted
- Yellow: between 5 and 10% of the enhancement proposals are accepted
- Red: between 2 and 5% of the enhancement proposals are accepted
- Black: less than 2% of the enhancement proposals are accepted

In order to compute the indicators above, the following measures are defined: *Number of enhancement proposals* and *Number of accepted enhancement proposals*. Furthermore, to be more explicit, in the Standard QualOSS Assessment Method, the type of data source to use when taking the measure and the type of artifact being measured must also specified as part of a measure name. Finally, every measure is associated to a measurement procedure (automated or manual or partly automated) to follow on in order to take the measure.

Definition of the Two Measures to compute the indicator above

Measure 1:

Base Name: Number of enhancement proposals

Data Source Type: issue tracking system

Artifact Type Being Measured: issue

Measurement Procedure:

Get the number of issues labeled as 'feature requests' or 'enhancements' that are reported on the issue tracking system

Measure 2:

Base Name: Number of accepted enhancement proposals

Data Source Type: issue_tracking_system

Artifact Type Being Measured: issue

Measurement Procedure:

Get the number of issues labelled 'feature requests' or 'enhancements' that are assigned to someone in the issue tracking system or that was not rejected (resolution is different from something like « Not a Bug » or « Won't fix »)

Measurement Status and Indicator Confidence Factor

In addition to computing the measure value and indicator value, the latest version of the QualOSS Methodology recommends to add a *status* to each measure value in order to specify if an actual measurement was performed as anticipated or if special circumstances arose during measurement. Furthermore, an indicator value should be further qualified with a confidence factor.

Measure Value

In the Standard QualOSS Assessment Method, it was decided to define Measurement Status and Indicator Confidence as follows.

Zero or positive status means that the measure may be considered. On the other hand, negative status means that the measurement could not be appropriately taken and should not be considered. The measure status and their meaning are as follows (note: below, measurement tools = automated or manual measurement procedures):

- **0** means measure value was taken on complete or nearly complete datasets and measurement tools work as expected on the datasets
- **1** means that datasets for the measure did not exist.
- **2** means that datasets were incomplete or noisy or that measurement tools did not work correctly on all datasets. However, the resulting measure value are reliable enough to be used in for indicators.
- **-1** means that a measure is not applicable in the context of the given assessment
- **-2** means that measurement tools did not work properly enough to provide reliable measure values
- **-3** means that a measure is not yet fully defined (in the current Std QAM version) or is of no interest to the assessment (based on stakeholder feedback)

Indicator Confidence

In Standard QualOSS Assessment Method v1.1 (latest version), Indicator Confidence [0, 1] is a composition of a importance weight [0, 1] and a measurement coverage [0, 100].

*Indicator Confidence = Importance Weight * Measurement Coverage / 100*

The I*mportance Weight* is used to quantify the added value of an indicator to answer a given question and transitively, how important an indicator is for assessing the risk for the given quality focus and the given viewpoint. Currently, the importance weights of the various indicator was calibrated based on the benchmark assessment found at http://ingrid.cetic.be:33323/qualoss_assessment. In particular, if an indicator could not be taken reliably for some of these assessment or if it seemed that the indicator did not truly reflect the risk situation for the given viewpoint and quality focus then its importance was lowered. For instance, the indicator above *Percentage of accepted enhancement proposals* is assigned the importance weight of 0.7,on the positive side it can reliably be computed for most F/OSS endeavor but on the negative side, the current thresholds of the indicator are still immature for the confidence to be higher.

The *Measurement Coverage* is obtained from the measurement status of the measures aggregated to form the indicator. In the simple model, is to simply compute the ratio of measures with positive status over the total number of measures in the indicator. However, in the Standard QualOSS Assessment Method v1.1, the developer of the indicator were given the freedom to replace this simple model with more complete conditional formulas.

The various computation of coverage can be found by exploring the spreadsheets of each assessment parts for the Asterisk assessment. They are reachable from the URL above and can be downloaded.

1.3.3.2 Executing the Standard QualOSS Assessment Method

The Standard QualOSS Assessment Method has been implement so that assessments can be conducted in a decentralized and modular fashion. In other words, instead of building a completely integrated platform as initially planned, it was decided to decouple the assessment from the collection of results in a centralized repository, which is also decoupled from the visualization tool developed to visualize assessment results.

The reason for this implementation is motivated from observing the advantages and disadvantages of existing solution to analyze F/OSS data, namely, QSOS (QSOS, 2006), OpenBRR (OpenBRR, 2005) and Ohloh (OhlohURL).

QSOS and OpenBRR have each developed method for assessing open source project. As further discussed in Section 1.4, their proposed solutions are quite light but fairly subjective and ambiguous. Nonetheless, they have shown that a decentralized assessment approach seemed appropriate. Their main tools are a spreadsheet and a form where an assessor does not need to be bug down with the installation of complex platform. On the other hand their weakness is the sharing of results and the shared visualization of results. With OpenBRR, spreadsheet visualization are proposed but no centralized repository of assessment results exists hence no one shares the OpenBRR results. This is somewhat different for QSOS which lately developed a client-side tool to view results. However, one must first download the client side tool and then download assessment results, to finally open these results with the client-side visualization tool.

Ohloh is the total opposite of QSOS and OpenBRR. It was not originally built to assess F/OSS projects but rather simply to view data and facts about F/OSS project. It is web-oriented as all data and derived facts are directly viewable through a web-browser. The main disadvantage of Ohloh however, is that it is unclear to how certain derived data and facts are obtained.

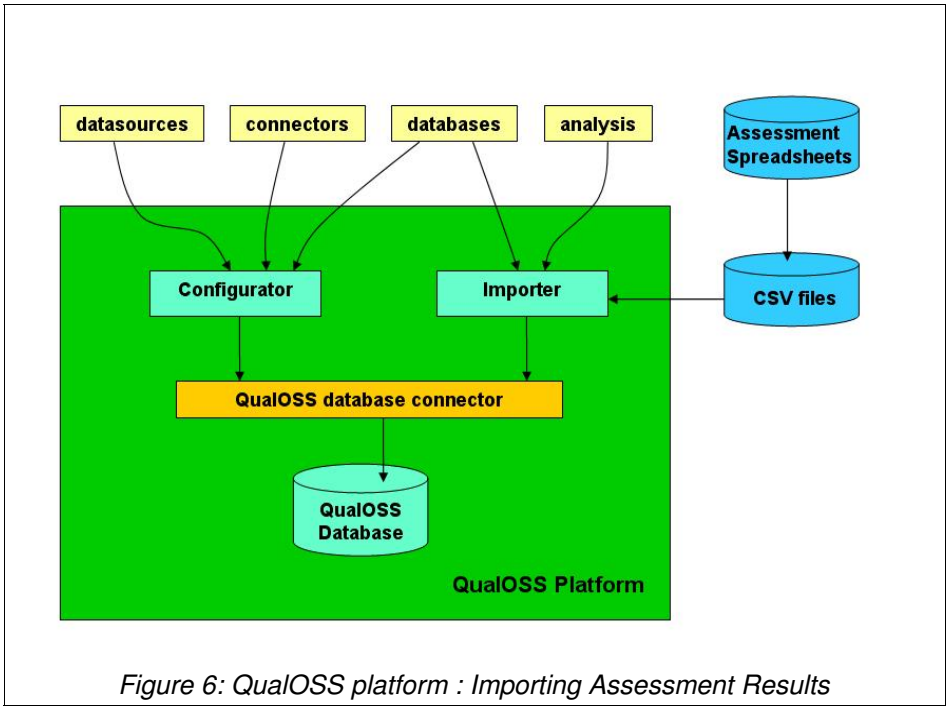

Figure 6: QualOSS platform : Importing Assessment Results

Learning from these two different approaches, QualOSS decided to decentralize assessment using spreadsheets as illustrated in the top blue cylinder of Figure 6. Given that the Standard QualOSS Assessment is more thorough than QSOS and OpenBRR, it was decided to split an assessment in various parts where each assessment part has its own spreadsheet. On the positive side, this split allows an assessment to be conduct in a modular fashion. For instance, two different people can assess different part of a F/OSS endeavor without overwriting risks. On the negative side, it creates a dangerous situation where the two people could end up with incompatible assessment results due to measures taken on datasets with different scopes, for example, the maintainability assessment only considers a small sub-component of a F/OSS project while the community assessment done by another assessor takes measure on the entire community of the F/OSS project. To avoid such conflict, the Standard QualOSS Assessment imposes a workflow where assessors are forced to synchronize their work to avoid incompatibilities in assessment results.

Once a spreadsheet for an assessment part is done, it can be uploaded to a centralized QualOSS repository using the QualOSS platform illustrated with Green box in Figure 6. Afterwards, one can then use the web visualization tool of the Standard QualOSS Assessment to view the assessment results as displayed in Figure 9. More information on the detailed assessment results are presented in the next section.

As mentioned previously, to avoid incoherent results, the Standard QualOSS Assessment Method imposes a detailed workflow. Figure 7 illustrates the operations of this workflow. The detailed actions of each operation are described in deliverable D4.5 of the QualOSS project.

In addition, the QualOSS Methodology imposes the assessment methods to guarantee traceability between input dataset, processing, and output results (intermediate or final). In the Standard QualOSS Assessment Method, v1.1, this traceability is provided in the form of LOG files. Thus, in parallel to filling spreadsheet for an assessment part, an assessor is also obligated to record the important actions in a LOG file. As for spreadsheet, each assessment part has its own LOG file. However, they follow a standard template with the following sections

1. Scoping Actions and Time
2. Computing Measures and Indicators
3. Summary
    1. Timing Information
    2. Deviation Summary
    3. Suggestion for Improvements
4. Additional Comments

Each assessment part is then free to refine this template to the specifics of that part of the assessment. The content to provide in each section of a LOG file is explained in the template LOG file provided as part of deliverable D4.5.

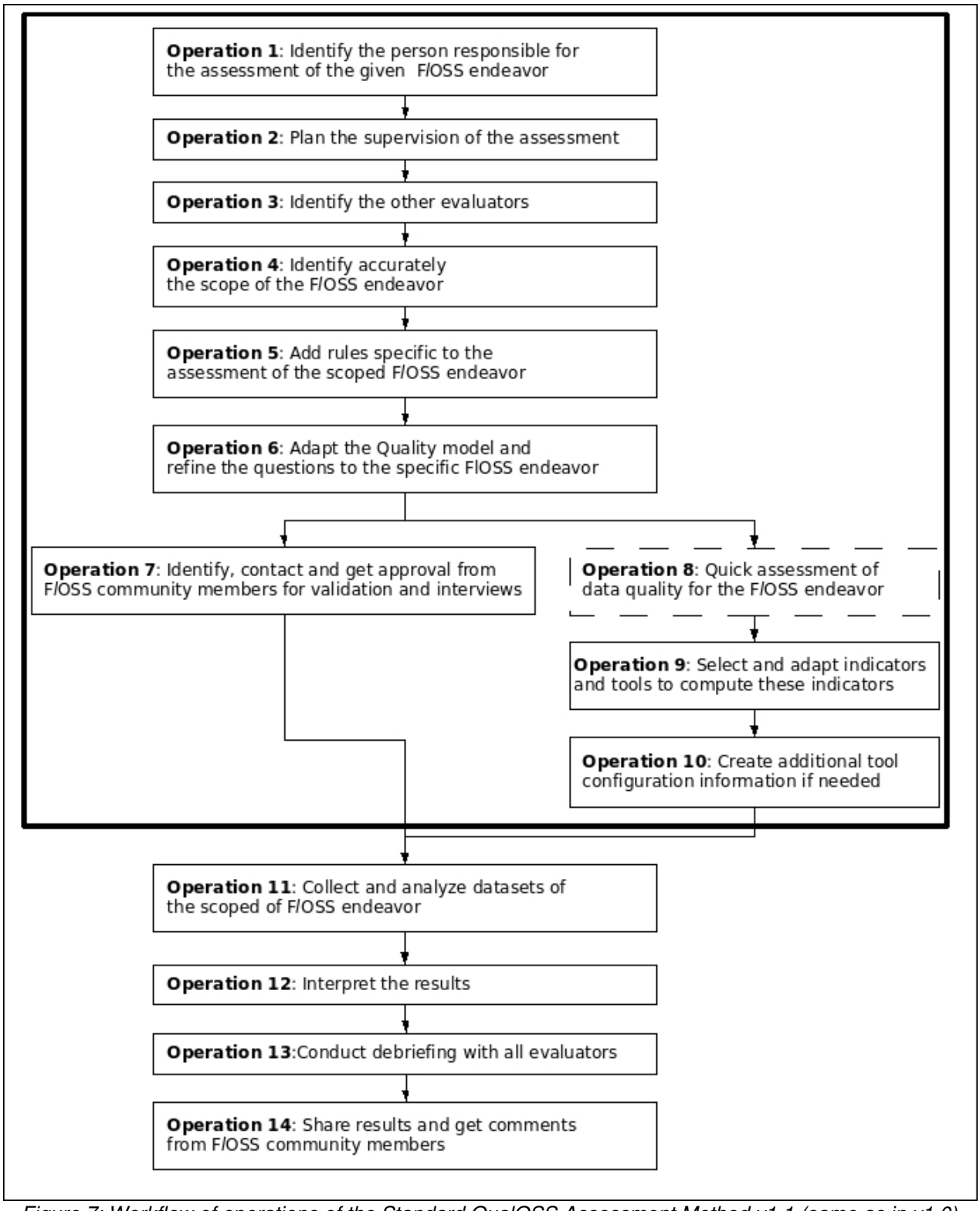

Figure 7: Workflow of operations of the Standard QualOSS Assessment Method v1.1 (same as in v1.0)

1.3.3.3 Assessment Results from the Standard QualOSS Assessment Method

This section provides an excerpt of assessment results obtained using the Standard QualOSS Assessment Method v1.1 executed on the Asterisk endeavor (core part only). The full version of the Asterisk assessment results are available from http://ingrid.cetic.be:33323/qualoss_assessment/ (under version 1.1).

The initial page of the Asterisk assessment results first remind the various item of the GQM template that were consider for the assessment, as illustrated in , the issue, the purpose, the context, the viewpoints, the broad quality focus and the F/OSS endeavor being assessed (including the version targeted for integration in a product)

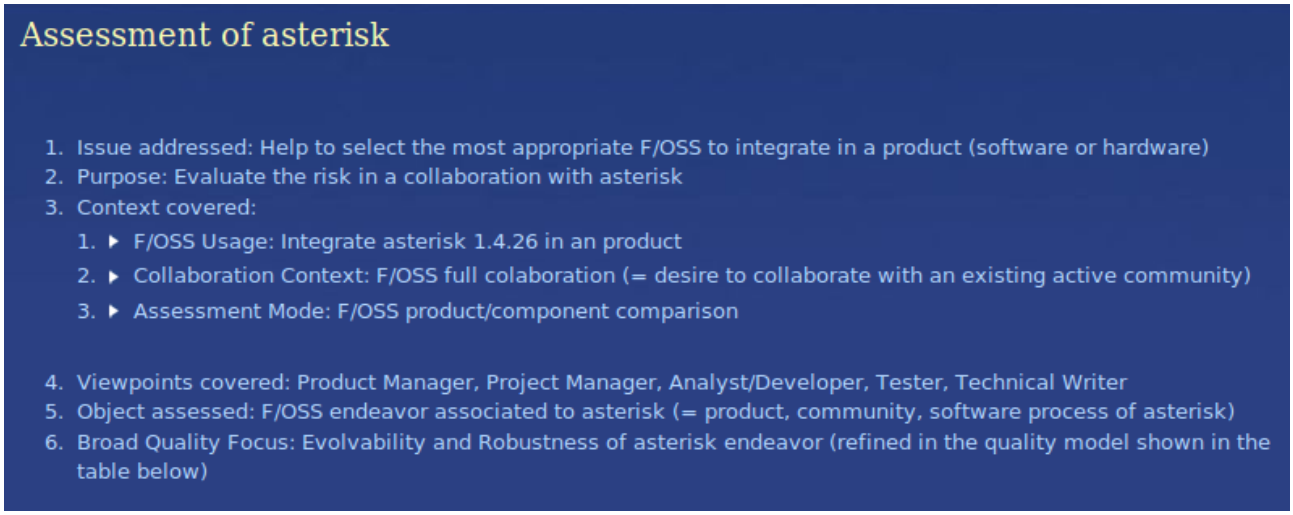

Figure 8: Information specifiying each element of the GQM high-level goal for a Standard QualOSS Assessment.

Below this reminder, the quality model of the Standard QualOSS Assessment is displayed with the aggregated risk scores. The reader can eventually decide to include or exclude certain quality attribute from the assessment.

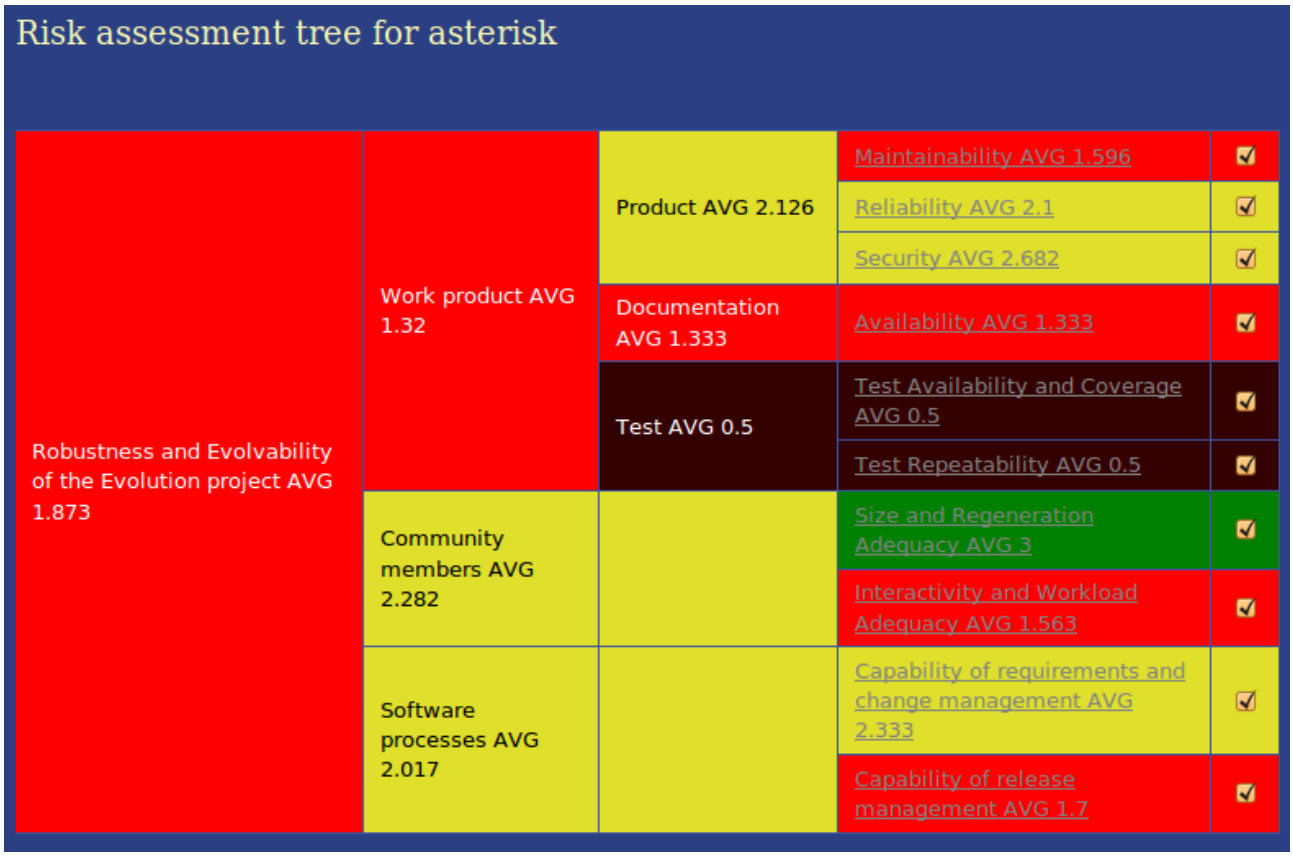

Figure 9: Overall assessment of Asterisk (targeted version 1.4.26) based on the 4-value scale (green, yellow, red, black).

This aggregation of risk score only reflects the average of the risk indicators from each of the quality indicators. This is fairly hard to interpret. For a more meaningful interpretation of the results, one can select the leaf of the tree to view the indicator specific indicators for each questions of each viewpoint. For the sake of brevity, only the page with the security assessment for product manager and project manager are illustrated in the next two figures, Figure 10 and Figure 11.

Figure 10 and Figure 11 both show the questions asked respectively by a product or project manager about security. The answers to these questions assess risk through the 4-color scale (green = no or negligible risk, yellow = small risk, red = medium risk, black = high risk).

In addition, each indicator is further qualified by a confidence factor. As mentioned in the previous section, the Standard QualOSS Assessment use a simple model to compute confidence by composing the important weight (how useful is the indicator for assessing the risk for the viewpoint and quality focus) and the measure coverage (how reliable are the measures used to compute the indicator value)

Although Industry users have commented that they would like to have the freedom to adapt the confidence factor, the QualOSS Methodology currently refuses that its assessment methods, in this case, the Standard QualOSS Assessment Method, grant that privileged to users. The main reason for preventing this right to is that a reader who did not perform the assessment may not fully grasp the concepts of importance weight and measure coverage. However, this restriction may be relaxed in the future.

Figure 10: Indicator color-based values and indicator confidence values for security related questions asked by the Product Manager (long term non-technical viewpoint)

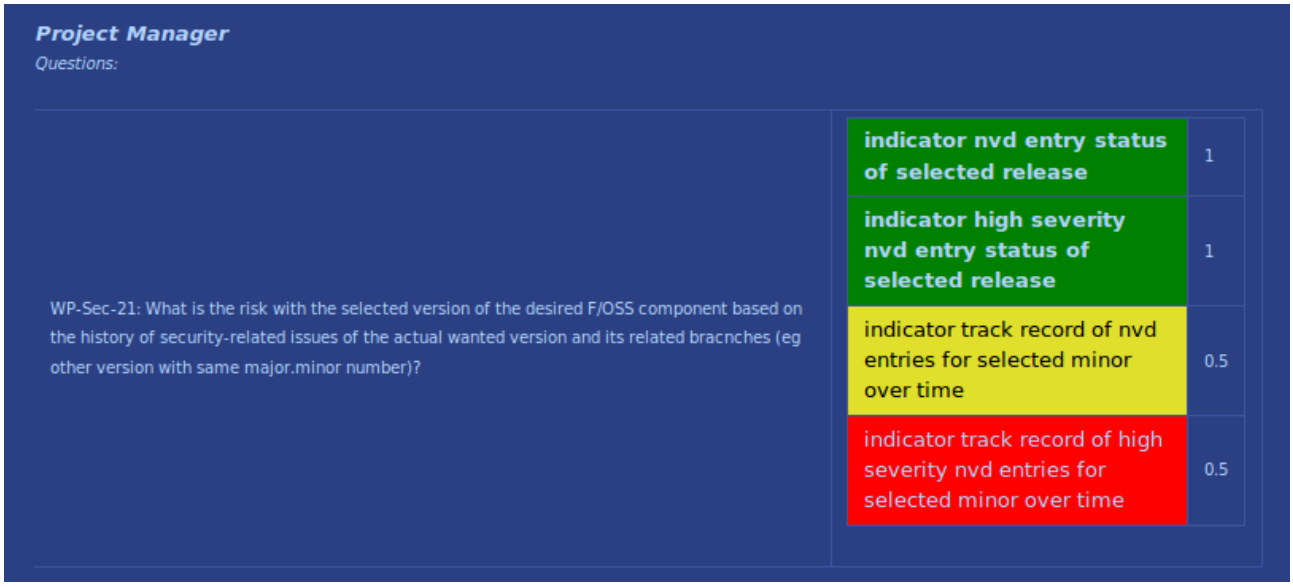

Figure 11: Indicator color-based values and indicator confidence values for security related questions asked by the Project Manager (short term non-technical viewpoint)

1.4 Comparison to Related Works

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 further developed below while research efforts in the latter category are only briefly mention in the next paragraph.

Several software quality assessment models and methods have been proposed through the years. A significant sample reveal the following works, the seminal McCall and Boehm models and assessment methods (Boehm et al., 1973, Boehm et al., 1976, Boehm et al., 1978, McCall et al., 1977), FURPS (Grady and Caswell, 1987, Grady, 1992), NASA SACT (Hyatt and Rosenberg, 1996), and ISO9126 (ISO 9126) and its new, upcoming version ISO25000 series to be published in the next 12 to 18 months. Furthermore, software process assessment method such as the Capability Maturity Model Integrated CMMI (Chrissis et al., 2006) or ISO15504 in combination with ISO12207 (ISO 15504, ISO 12207). Unfortunately, all these assessment methods cannot be applied for F/OSS assessment because they require a deep control of the software development process where the various actors are required to perform certain actions and to collect certain data, for example, conduct review of specifications to identify errors, conduct well defined verification to compute meantime between failures, etc. Various F/OSS endeavors do not follow the same approach and do not collect or share such data. Furthermore, F/OSS usually do not go through intensive design phases and rarely propose specification document. In most cases, F/OSS endeavor are started by few developers who have a shared vision and directly implement this vision. Subsequently, the source code of the initial version becomes the only model of the software and specification documents are not created. Thus, preventing the application of the the traditional assessment methods mentioned above. Although not applicable to F/OSS, these approaches above still influenced how the Standard QualOSS Assessment Method was built based on a quality model and an assessment method to evaluate the various quality attributes inventoried of 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. FLOSSMETRICS, 4. FLOSSMOLE, 5. OHLOH

First a comparison of QualOSS with QSOS and OpenBRR is presented and then connections with the other 3 data collecting projects are reviewed.

OpenBRR (OpenBRR, 2005) and QSOS (QSOS, 2006) propose a fairly similar approach for assessing F/OSS. A detailed comparison of the two assessment methods is presented in (Deprez, 2008). One important difference between these two methods lies in the fact that QSOS focuses its assessment on a version while OpenBRR does not target any particular version. For the Standard QualOSS Assessment Method, the QSOS approach was followed where a particular version targeted for integration (in a product) must be identified in order to start an assessment. QSOS and OpenBRR are both lightweight in that they propose simple manual scoring procedures that most IT people can easily follow. However, with over

simplicity comes ambiguity. Many of the scoring criteria in both methods can be considered ambiguous. In particular, different people would likely quite interpret the scoring procedure differently, which may drastically influence the assessment results. One of the main goal of QualOSS is to eliminate the influencing source of ambiguity in how assessment obtain scores.

The main difference between QualOSS and these two methods is that QSOS and OpenBRR believe that a single assessment method is fit to assess all F/OSS integration context. However, the QualOSS methodology believes that an assessment goals become different depending on context of the F/OSS integration. As presented previously, enterprise would have different goals if they plan to integrate a F/OSS component in a product, in a service or in an infrastructure. Furthermore, this difference in assessment may also be impacted by the fact that an enterprise plan to collaborate or not with a F/OSS endeavor or if the assessment is done in the purpose of a component or a version comparison. This difference in assessment goals would then explicitly show in that the various viewpoints could be different and more importantly, different questions would be asked by these various roles.

As a result, the QualOSS approach is first to propose a overarching QualOSS methodology that one can follow to derive an assessment method specific for the assessment of a given context. Clearly, some of the questions, viewpoint may be of interest to several F/OSS integration context. However, until questions and measures are explicitly identified as being a shared concerned across several F/OSS integration contexts, QualOSS does not recommend sharing assessment results for other F/OSS integration contexts than the one for which the assessment method was targeted. In other words, the Standard QualOSS Assessment Method was built for F/OSS integration in a product, F/OSS full collaboration, and component comparison. Consequently, it is not recommended to use the Standard QualOSS Assessment Method for version comparison, for F/OSS fork or for integration in a service, at least until the Standard QualOSS Assessment Method has been validated for these other contexts.

In contrast, QSOS and OpenBRR propose to apply their methods to all different integration contexts. Consequently, this forces them to remain fairly ambiguous in what they are assessing and decreases the reliability and the value addition of their assessment results.

QualOSS however, is currently a bit heavier in its application than QSOS or OpenBRR. Where QSOS and OpenBRR take roughly 1 person-day, the Standard QualOSS Assessment Method currently takes between 3 to 5 person-days. This increase is mainly due to two reasons. First, to increase objectivity the QualOSS Methodology requires using the GQM. Consequently, tools to take measures must be applied. Their proper application requires a thorough gathering process of the input dataset. Thus, the data preparation and in some cases, the measurement process require more time than the simple scoring criteria from QSOS and OpenBRR. Second, the QualOSS methodology requires that assessment results should be traceable. In other words, traceability links must be kept between input datasets, processing and output results (intermediate and final). In the case of the Standard QualOSS Assessment Method, the traceability requirements is satisfied by the assessor who must fill a LOG file for each assessment part.

It is worth noting that the traceability requirement is one of the mechanism put in place by the QualOSS Methodology to remove ambiguity in assessment results. This is yet another difference between the other two assessment methods and QualOSS. Actually, the latest QualOSS visualization tool also includes links to LOG files that maintain the traceability on the input datasets used for measurement, the measurement procedures and intermediate results. Additional traceability information including final measure and indicator results are stored in the assessment spreadsheets.

Thus, if one follows the traceability requirement imposed by the QualOSS Methodology, anyone consulting the results can drill down to the details of how measurements were taken and verify whether or not they find erroneous or questionable results.

Besides QSOS and OpenBRR, 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 its ability to identify the F/OSS license present in source code files. This information would help to compute risks related to conflicts in F/OSS licenses within a F/OSS endeavor. Subsequently, this could be used for measuring F/OSS endeavor compatibility, which is part of the quality model of the Standard QualOSS Assessment Method but for which no indicators have been developed yet. Finally, it is worht reminding that Ohloh influenced the architecture of the QualOSS framework where it was decided during the last year of the project to propose a web interface to view the QualOSS assessment results directly through a web browser rather than download them in the form of a pdf report.

2. Dissemination and Use

QualOSS disseminated its work through different channels, in particular, research publications and industry-oriented events. Over its whole duration, QualOSS partners published a total of 27 research papers in various conferences. Furthermore, ZEA continuously relayed QualOSS status to the Plone community during Plone world conferences. Importantly, CETIC and IESE interacted with standardization bodies and presented QualOSS findings. In particular, CETIC is a Belgian representative in ISO/IEC-Subcommittee 7 that produces standards related to Software and Service Engineering. CETIC actively participated to the ISO meetings twice a year where they presented selected QualOSS outcomes to representatives of other national body in two relevant working groups. IESE also interacted with the German Pharmaceutical Association to advertise F/OSS and QualOSS.

QualOSS was also involved in the organization of 5 F/OSS events, *i.e.*, 3 occurrences of the Workshop on Public Data about Software Development (WoPDaSD) in 2007, 2008 and 2009 and 2 occurrences of the Workshop on "libre software research meets libre software developers" as a part of FOSDEM in 2007 and 2008.

Besides these dissemination actions, QualOSS partners have displayed the practicality of QualOSS results in 4 industrial case studies. Each case study evaluated the user satisfaction and the profitability of QualOSS assessment results. A brief description of these case studies is provided in Section 2.1.1. Afterwards, Section 2.1.2 gives a general exploitation plan for consultant.

2.1.1 Case Studies

The 4 cases studies perform during the QualOSS project explored the user satisfaction and profitability of assessment results obtained using the Standard QualOSS Assessment Method. These case studies involved various F/OSS integration contexts. Thus, with these case studies, the Standard QualOSS Assessment Method was studied beyond the context for which it is was built. As a reminder, the Standard QualOSS Assessment Method was specifically built for the following context:

- F/OSS Usage Context: Integration of F/OSS code in a software product.
- F/OSS collaboration context: F/OSS full collaboration where an enterprise want to establish a long lasting relationship with a F/OSS endeavor.
- F/OSS Assessment Mode: Product Comparison

A brief but more detailed description of the studies are found in the next subsection, however, it is worth mentioning that only the AdaCore/Coverage tool matched the context above. In the AdaCore/Gcc-backend study, the assessment mode is version comparison and not product comparison, in the OSL/Yanolc study, the F/OSS collaboration context is F/OSS fork and not F/OSS full collaboration, and in the Freecode/Asterisk study, the F/OSS usage is integration in a service and not in a product. Thus, it is expected that the assessment results will not always fulfill the stakeholders' needs in all case studies. Their goal was to determine what work and what did not well with the Standard QualOSS Assessment Method in these various F/OSS integration contexts.

It is worth highlighting that 2 out of the 4 case studies involved companies outside the QualOSS consortium, namely, OSL and Freecode that willingly dedicated time for interviews. The other two case studies involved AdaCore who is directly involved in the QualOSS project.

2.1.1.1 AdaCore/Gcc-backend Study

A first case study involved the assessment of GCC-backend to help AdaCore substantiate upgrade decision (or not) to a newer version of the GCC-backend in their Ada compiler toolsuite named GNAT Pro.

The context of this case study was

- F/OSS Usage Context: Integration of F/OSS code in a software product.
- F/OSS collaboration context: F/OSS full collaboration where AdaCore has already established a long lasting relationship with Gcc-backend and plans to continue that relationship.
- F/OSS Assessment Mode: **Version Comparison**

The part of the context in bold indicates a mismatch compared to the context for which the Standard QualOSS Assessment was build. In this cases, the AdaCore/Gcc-backend is a version comparison of Gcc-backend where versions 4.2 and 4.3 were assessed separately and then the two results were compared. AdaCore already had the prior knowledge that the 4.2 version introduced significant new functionality which weakened the level of reliability of Gcc-backend. Consequently, they wanted to verify what indicators in the QualOSS assessment results would also identify this fact.

Incidentally, the version of the Standard QualOSS Assessment Method used for this case study is *v1.0_RC*

In summary, AdaCore found that code assessment results were quite useful to confirm their expectation. This study also revealed that no test coverage reports currently existed. On the other hand, given their current active involvement in the Gcc-backend endeavor, AdaCore did not believe that assessment of documentation, community and software processes provided useful information in their context since they already knew that information and the assessment results did not teach anything new.

This level of satisfaction was expected as the assessment mode is version comparison and not product comparison. Currently, the questions and indicators for assessing community and software processes are not adapted for version comparison. They actually provide the same assessment results for version 4.2 and 4.3. With regards to documentation, AdaCore already knew the quality of documentation. Since Gcc-backend evolves and plans its releases in a systematic fashion, documentation can be reused from one version to the next one and changes to documentation are usually moderate between Gcc versions.

AdaCore is now planing to use some QualOSS indicators to verify the readiness of new Gcc-backend code for integration in their GNAT Pro compiler. During a debriefing interview with AdaCore, it was asked if it would be useful to adapt the current assessment of community, software processes and documentation to better address their context. Although new assessment questions were identified as interesting during these interview, AdaCore recognized that given their intensive collaboration with the Gcc-backend community, community assessment would not probably required too much effort to identify new information. With regards to software processes, Gcc-backend has a fairly well established process thus again little would be gained from a fine grain assessment to identify differences in software processes between versions. A similar argument applies to documentation.

In conclusion, AdaCore will simply apply code assessment indicators to substantiate their migration decision but they will not assess or use other part of the quality model of the Standard QualOSS Assessment Method.

2.1.1.2 OSL/Yanolc Study

A second case study involved the assessment of Yanolc (client-side) in order to help Océ Software Laboratories (OSL), a company in Belgium, to determine if it would implement an lpr client (client used to communicate with a printing device) using existing internal components or using the Yanolc client-side code instead. It is currently estimated that a combined effort of 2 to 3 PM would be required to integrate existing internal components into a polished lpr client. Thus, Yanolc would only be considered if the anticipated effort remains around the same estimation.

The context of this case study was

- F/OSS Usage Context: Integration of F/OSS code in a software product.
- F/OSS collaboration context: **F/OSS fork** where OSL would fork a version of Yanolc and start their own development project based on it without planing any further collaboration with Yanolc in the future.

- F/OSS Assessment Mode: Product Comparison (where Yanolc is compared with the alternative of integrating existing internal components

The part of the context in bold indicates a mismatch with the context for which the Standard QualOSS Assessment was build. In this case, OSL follows processes from the Océ Headquarter which are not familiar with potential benefit of collaborating with F/OSS endeavors. Consequently, until their internal development process allows for such collaboration, they can only get involved in F/OSS fork when they want to leverage an existing F/OSS code. OSL will distribute the lpr client to their customer hence in case of bugs, they need to have complete understanding of the application. Consequently, OSL is mostly interested in the work product assessment, more specifically, code and documentation. For Test, they already have an large internal testsuite built from the lpr RCF. A priori, OSL seemed less interested in the assessment of community and software processes.

Incidentally, the version of the Standard QualOSS Assessment Method used for this case study is *v1.0_RC*

In this case, OSL found the assessment of maintainability and documentation useful. The Standard QualOSS Assessment answered relevant questions to help them make a decision. OSL noted that many questions could not be answered because no bugtracker data was available. Since many indicators are based on measures on a bugtracker dataset, a significant number of measures could not be taken during the assessment. OSL viewed this lack of information as a risk. Thus, they agree that they should be indicated with black indicators. In the end, OSL decided to implement the lpr client using the internal components.

Surprisingly, OSL thought that they would not be interested in community assessment. However, a factor that heavily influenced their decision was that Yanolc was a 1-man effort. This information made OSL feel that they would not benefit from the support of a large community and thus, they saw using Yanolc as highly risky due to this small community.

In conclusion, OSL found the results of the QualOSS assessment informative even in the cases where measures could not be taken due to the lack of input data, for instance, bugtracker missing. Regarding community and software processes, a very light assessment would also bring relevant information but in their F/OSS fork collaboration context, they did not need very details information and only computing a few indicators on these two topics would capture sufficient information. Finally, OSL also noted that they would be quite interested in indicators on the compatibility (or incompatibility) of F/OSS licenses used by a F/OSS component and second, information about the firms involved in a F/OSS endeavor, in particular, to know about the participation (or not) of their direct competitors in a F/OSS endeavor. These indicator would respectively normally be found under the Tools and Dependencies – F/OSS endeavor compatibility and under Community – Composition Adequacy, unfortunately due to lack of time, the creation of indicators for these two characteristics was put on hold until future releases of the Standard QualOSS Assessment Method.

2.1.1.3 AdaCore/CoverageTool Study

A third case study involved the assessment of a Coverage tool implemented by AdaCore and a consortium of other companies. In this context, AdaCore, leader of the project, plans to release the Coverage tool under an F/OSS license. Consequently, they want to determine its readiness for an F/OSS releases.

The context of this case study is

- F/OSS Usage Context: AdaCore performs an introspection of the Coverage tool endeavor to see its readiness for F/OSS release. However, AdaCore assumes that external companies that would use the Coverage tool would do so by either integrating it in a product or in an infrastructure.
- F/OSS collaboration context: F/OSS full collaboration where AdaCore hopes that people external do AdaCore will contribute and establish a long lasting relationship with Coverage Tool endeavor.
- F/OSS Assessment Mode: Product Comparison (where the results of the QualOSS assessment could later be used to compare Asterisk to other similar F/OSS endeavor)

This context matches exactly with the one expected for the Standard QualOSS Assessment Method. The main difference is that the assessment results are used for an introspection, that is, used by AdaCore, the developers of the Coverage tool and not by an enterprise who is planning to integrate the Coverage tool in one of its product.

Incidentally, the version of the Standard QualOSS Assessment Method used for this case study was *v1.0*.

AdaCore members who participated to QualOSS performed the QualOSS assessment. Afterwards, they had some very positive comments regarding the test and software process assessment results. Based on these results, AdaCore identified high-priority actions to improve their risk indicators for these two parts of the assessment. Although they also found the results of the documentation quite useful, no actions have been planned in the short term, as they believe that the current documentation is sufficient for a young project. Finally, the product and community assessment results were not perceived as useful because they were not well adapted for very a young project. The product part of a QualOSS assessment currently assumes that a F/OSS component has a certain lifespan so several stable releases have been made available. This is needed to study bugs evolution and also to study how much changes happenned on the code base between stable versions. For young project, without several stable releases whose code base quickly evolves, maintainability and reliability indicators were mostly black (high risk) either because measure results showed risky or because measures could not be taken at all. Although this is picture may not be pleasing for young F/OSS project, it still seems an appropriate assessment results since an external company would consider code contribution in a volatile code base as a risk at this early stage. An improvement of the maintainability and reliability assessment could be to allow alternate measures to be taken on the source code management (version control) tool instead of on packaged distributions. In this case, the “stable” releases considered for assessment could have been decided with the help of the project responsible. Similarly for community assessment, AdaCore members are the only developers at the moment, external users have started to report bugs and enhancement requests but have not contributed to the code yet. Thus, community assessment results show a risky situation. Again, although not pleasing, these results seem adequate since external companies would find dealing with a small community controlled by a single company as fairly risky.

On the positive side, AdaCore thought that by increasing their indicator score on test and software processes, they would automatically increase their chances to develop a thriving community of external contributors. However, AdaCore is also fairly commited to this tool and will in any case continue developing and supporting it even without contribution of external people.

2.1.1.4 Freecode/Asterisk Study

A forth case study involved the assessment of Asterisk for Freecode, a Norwegian open source integrator. Currently, Freecode uses OpenBRR for assessing F/OSS however they are not satisfied with it and would consider switching to a new assessment method. Since Freecode already assessed Asterisk with OpenBRR, they wanted to compare QualOSS results with those of OpenBRR.

The context of this case study is

- F/OSS Usage Context: **Integration of F/OSS component in a service**.
- F/OSS collaboration context: F/OSS full collaboration where an enterprise want to establish a long lasting relationship with a F/OSS endeavor.
- F/OSS Assessment Mode: Product Comparison (where the results of the QualOSS assessmetn could later be used to compare Asterisk to other similar F/OSS endeavor)

The part of the context in bold indicates a mismatch with the context for which the Standard QualOSS Assessment was build. In this case, Freecode does not plan to integrate Asterisk in one of its product but rather to sell configuration services to third parties. Freecode is already an advanced user of Asterisk and they already know the weaknesses of this F/OSS endeavor, in particular, it is controlled by a single company that rarely accepts contributions from others. This creates problems for Freecode that patches certain

security bugs in Asterisk which are not integrated in the main Asterisk code base. This forces Freecode to keep an internal copy of its patches and to apply them to each newer release prior to installing it at its customers' site. Given this knowledge, they wanted to know if by any chance, some QualOSS indicatos would have directly of indirectly catch this detail.

The version of the Standard QualOSS Assessment Method used for this case study was *v1.1*.

Freecode had a dual reaction regarding assessment results from the Standard QualOSS Assessment Method. First, they thought that the large number of measures was overwhelming. However, on second thoughts, after having reviewed the questions and risk indicators, they agreed that they were all fairly useful and answers worthy questions. This feeling about the quantity of measures was due to the fact that initially, the assessment results were presented directly from the spreadsheets, however, once the results have been loaded in the QualOSS repository and viewable through the QualOSS visualization tool, they felt much more at ease and were interested by these results.

One negative comment regarded the lack of assessment for functionality. Clearly, assessing functionality would be interesting. However, such an assessment would only be useful if it was specific to a type of software, for instance, in this Asterisk case, it would be interesting to know all the user functions available in VOIP systems in general and then measure how many of them are present in Asterisk. Unfortunately, identifying the exhaustive list of functionality for each software type is practically infeasible, even for a given family of software such as all VOIP systems, this exercise is tedious. Furthermore, QSOS has already started such an effort for a few family of software such as database servers, CMS, etc. Thus, instead of redeveloping a very similar approach, it seemed more appropriate to point Freecode the functionality assessment of QSOS. Actually, QSOS would welcome Freecode's contribution of a list of VOIP system functionality.

During this case study, the QualOSS members performed the QualOSS assess for Asterisk thus, Freecode did not performed the Standard QualOSS Assessment Method but only commented on the assessment results. Consequently, they needed further investigation on the effort needed to learn and then conduct assessment in order to decide if they will use QualOSS in the future. QualOSS partners will therefore continue communicating with Freecode to also show them how to eventually create new questions, indicators and measures. This collaboration outside the QualOSS project could yield to the development of other QualOSS Assessment Methods more appropriate to other F/OSS integration contexts, for example, one specific for the integration in a service context.

2.1.2 Exploitation scenarios for consultant

Beside the case studies conducted during the QualOSS project, more generic exploitation scenarios are envisaged. To present them, it is first useful to list the project results which can be used for building consultancy services.

The QualOSS results that can be exploited to build consultancy services are

1. The QualOSS Methodology
2. The Standard QualOSS Assessment Method (and its associated assessment tools)
3. The QualOSS platform including visualization capability
4. The QualOSS repository of assessment results (using the Standard QualOSS Assessment Method)

CETIC currently has an instance of the QualOSS platform running on a public URL (http://ingrid.cetic.be:33323/qualoss_assessment/) where assessment results contained in the QualOSS repository can be viewed. The repository will continue to be selectively enriched with new assessment results. CETIC will take the opportunity offered by the project CELLaVI, funded through European Structural funding, to perform new Standard QualOSS Assessment and add them to the QualOSS repository. This

presentation of QualOSS results will server as free advertisement to demonstrate the capabilities of the Standard QualOSS Assessment Method.

Below, a list of generic enterprise needs related to F/OSS assessment are described and then a short explanation is given on how the QualOSS results listed above can be used to solve each need. An estimation of the effort needed to deploy the proposed solution is also given.

F/OSS-based Software Development Project

This first scenario involves an enterprise that wants the development of a F/OSS-based software product. This product can either be developed in house or by a third party. The goal is to reuse F/OSS code wherever possible. For certain functionality, several F/OSS components will compete. It is therefore important to make an objective judgement for selecting the best F/OSS endeavor to collaborate with rather than merely using the first F/OSS component encountered. Even in cases where only a single F/OSS component exists, one always needs to make a informed decision on whether to use that component or reimplement the needed functionality from scratch or from lower level F/OSS components.

The Standard QualOSS Assessment Method fits extremely well with this context. Since it addresses that very context: integration in a product with F/OSS full collaboration anticipated where several product are compared. An overall operational approach for F/OSS endeavor selection in this context is presented in (Majchrowski, 2008). The Standard QualOSS Assessment Method could be used for each selection exercise encountered during this overall F/OSS integration process.

The effort needed for a thorough application of the Standard QualOSS Assessment Method is between 3 and 5 person-days per assessment. However, it is always possible to agree on a lighter assessment still acceptable for the given context. It is however worth noting that the operational approach suggested is to perform a first filter and only perform a complete QualOSS Assessment for only the first two, maximum three F/OSS components and endeavors of interest.

With regards to the applicability of QualOSS for this scenario, it is worth mentioning that even prior to the QualOSS project, CETIC started advising customer in F/OSS selection using the operational approach above and the QSOS assessment method. In the future, QSOS will be replaced by a QualOSS Assesment Method in order to obtain more objective results.

F/OSS Assessment Program

This scenario is envisaged for companies who want to establish a F/OSS selection program for their whole enterprises. To estimate the appropriate effort to create such an F/OSS assessment and selection program, first, the various integration contexts targeted must be identified. Then the tailoring of the QualOSS Assessment Method needed to address each context must be estimated. Finally, the effort for conducting an assessment should also be projected.

Depending on the contexts targeted, modifications to be implemented in the Standard QualOSS Assessment Method will vary between 2 person-days to possibly, 1 person-month (or even more) if new measures and new measurement tools must be implemented. However, it is possible to phase the adaptation of the Standard QualOSS Assessment Method to fit to the newly targeted integration context on a need-to basis.

Once the new QualOSS Assessment Method is developed, it also will take time to perform each assessment. The time to execute an assessment will automatically vary based on the selected QualOSS Assessment Method. As an indication, the Standard QualOSS Assessment Method currently takes between 3 and 5 person days. In the case of frequent F/OSS assessments, it is probably worth investing time augmenting the automated part of an assessment and to hook the necessary connectors to the QualOSS platform in order to activate the automatic computation of certain measures and indicators.

In terms of the usabability of the QualOSS results, this scenario requires using the QualOSS Methodology as a guide to perform acceptable tailoring to the Standard QualOSS Assessment Method. Second, the Standard QualOSS Assessment Method, can be used as a starting point to generate ideas for new measures, indicators, questions, roles, quality focuses or contexts. Finally, for a F/OSS assessment program

where frequent assessments are expected, it would likely be a worthy investment to install and run an instance of the QualOSS platform within an organization.

The installation of the QualOSS platform requires between 0.5 and 1 person-day. However this number may grow if additional customization (new measures, measurement tool. indicators, etc.) must be integrated in the QualOSS platform. CETIC is currently exploring with certain of its existing client organizations the possibility to implement this scenario.

Training for New Assessors

Training on the various QualOSS results can also be an interesting way to become independent for an enterprise. Currently, it is anticipated that a course covering the QualOSS Methodology, its Standard QualOSS Assessment Method, and topics on how to create new QualOSS Assessment method would require a combined effort of 1 to 2 person-days. A second day gives ample time to cover examples.

A training on the QualOSS Platform is also possible. It is also anticipated to last between 1 and 2 person-days. The second day allows covering automation of measurement and the addition of new measurement tools to the QualOSS platform.

CETIC has initiated the contact to become a Qualipso competence center for the Benelux region. Once the training material has been developed, CETIC will be able to export to other competence center through out the world.

3. 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.

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

(Boehm et al., 1973) B.W. Boehm, J.R. Brown, H. Kaspar, M. Lipow, G.J. MacLeod, and M.J. Merritt. Characteristics of Software Quality. TRW Software Series TRW-SS-73-09. December, 1973.

(Boehm et al., 1976) B. W. Boehm, J.R. Brown, and M. Lipow. Quantitative evaluation of software quality. In Proceedings of the 2nd International Conference on Software Engineering (ICSE) 1976, pages 592-605, 1976.

(Boehm et al., 1978) Barry W. Boehm, John R. Brown, Hans Kaspar, Myron Lipow, Gordon J. MacLeod, and Michael J. Merritt, Characteristics of Software Quality. North Holland Publishing Company, 1978.

(Chrissis et al., 2006) Chrissis, Mary Beth; Konrad, Mike; Shrum, Sandy (2006). CMMI: Guidelines for Process Integration and Product Improvement (2nd Edition). Addison-Wesley Professional. ISBN 0321279670. http://www.sei.cmu.edu/library/abstracts/books/0321279670.cfm.

(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.

(Majchrowski, 2008) Annick Majchrowski and Jean-Christophe Deprez, *An Operational Approach for Selecting Open Source Components in a Software Development Project,* EuroSPI, Communications in Computer and Information Science, Vol. 16, pp. 176-188, Springer, 2008.

(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.

(Grady and Caswell, 1987) Robert B. Grady and Deborah L. Caswell, Software Metrics: Establishing a Company-Wide Program, Prentice-Hall, 1987

(Grady, 1992) Robert B. Grady, Practical Software Metrics for Project Management and Process Improvement. Prentice Hall, 1992

(Gartner, 2008) http://www.gartner.com/it/page.jsp?id=593207, published on Janurary 31, 2008.

(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.

(Hyatt and Rosenberg, 1996) Larry Hyatt and Linda Rosenberg, A software quality model and metrics for risk assessment, in Proceedings of the ESA 1996 Product Assurance Symposium and Software Product ASS, 1996.

(McCall et al., 1977) J.A.McCall, P.K.Richards, G.F.Walters, "Factors in Software Quality", RADC TR-77-369, 1977. Vols I,II,III', US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055, 1977.

(ISO 9126) ISO/IEC 9126 International Standard, Software engineering – Product quality, Part 1: Quality model, 2001.

(ISO 12207) ISO/IEC 12207: International Standard, Information technology -- Software life cycle processes, 1995

(ISO 15504) ISO/IEC 15504: International Standard, Information technology Process assessment, 2004.

(OhlohURL) http://www.ohloh.net/,

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

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