Sponsored through Framework Programme Sixth (Call 5) by

Document Information

Version: 1.1
Date : Jan 13, 10
Pages : 33

Owning Partner:
AdaCore

Author(s):
José Ruiz (AdaCore)
Jacques Flamand (CETIC)
Ruediger Glott (MERIT)

Reviewer(s):
José Ruiz (AdaCore)
Olivier Ramonat (AdaCore)
Jacques Flamand (CETIC)
Ruediger Glott (MERIT)

To:
European Commission

Purpose of distribution:
Final Submission

The QUALOSS Consortium consists of: CETIC (BE), Facultés 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)

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: D5.2

Title:

Measurements Report of Case F/OSS projects

Deliverable: D5.2

Title: Measurements Report of Case F/OSS projects

Executive Summary:

One of the most important aspects in QualOSS is the validation of the QualOSS methodology and of the QualOSS assessment methods developed in the project, with a particular emphasis, in the initial case studies, on the validation of the standard QualOSS assessment method. To achieve such a validation, a set of case studies are devised to verify whether or not particular business goals are reached. A set of suitable pilot projects are identified to assess the applicability and utility of the QualOSS methodology and QualOSS methods.

People directly involved in the pilot projects will be interviewed to better understand the general context in which the QualOSS methodology and QualOSS assessment methods will be applied. These interviews will help to verify several hypotheses regarding user satisfaction and profitability. First, they will allow for comparing the results obtained from QualOSS assessments against human perception of the robustness and evolvability of the F/OSS endeavors assessed in each case study. Second, these interviews are also useful to study user satisfaction with the results obtained from the standard QualOSS assessment method and eventually of other more advanced QualOSS assessment methods. The focus in this document is set on the evaluation of indicators and the human perception of the assessment, while overall user satisfaction with the measurement results will be dealt with explicitly in D5.3.

TABLE OF CONTENTS

1. Introduction

1.1 Motivation

The strategic objective of the QualOSS project is to enhance the competitive position of the European software industry by providing methodologies and tools for improving their productivity and the quality of their software products. To achieve this objective, QualOSS notes that many organizations integrate Free *libre* Open Source Software (F/OSS) in their systems hence QualOSS goal is to facilitate the identification of the most robust and evolvable F/OSS development endeavors whose F/OSS components are worth integrating in industrial software products and systems. In the end, the QualOSS methodology and its QualOSS assessment methods will ease the selection of high quality open source components. The overall effect will be increased productivity and higher dependability for the industrial software products integrating F/OSS components.

To achieve this goal, QualOSS proposes to build a high level methodology to benchmark the quality of open source software. In particular, the QualOSS project delivers an assessment methodology for gauging the evolvability and robustness of open source software endeavors.

This fifth work package (WP5) verifies that the QualOSS methodology, its QualOSS assessment methods and the accompanying tools can be used to verify whether particular business goals set for the studied projects are reached or not. The first task of WP5 (T5.1) presents the broad context of the case studies and selects the pilot projects to be analyzed. Furthermore, T5.1 lists the hypotheses that will be checked by each case study. The second task (T5.2) of WP5 consists in a set of interviews and the application of QualOSS assessment methods on the selected FlOSS endeavors. In a first phase, T5.2 applies the standard QualOSS assessment method on the FlOSS endeavors. In a second phase, the adaptation of the standard QualOSS assessment method into more advanced methods and the application of these advanced methods will be studied. The results obtained from T5.2 are used in the final task (T5.3) to report the results of the case studies and to argue the validity of the hypotheses being tested.

1.2 Objectives

The goal of task 5.1 was to design the case studies and find the pilot projects on which these studies can be conducted. It included the identification of the hypotheses to test, the design of the general protocol to use in case studies, and the description of the pilot projects on which these studies are to be conducted.

Task 5.2 measures quality characteristics, indicators and metrics required to validate these hypotheses. People directly involved in the pilot projects are then interviewed, giving them access to the results of the QualOSS assessment, to assess the human perception of the assessment and allow for the validation of hypotheses in Task 5.3.

1.3 Approach

This document contains the measurements related to metrics and indicator for the selected pilot projects. It provides also the summary of the interviews with people directly involved in the projects.

The people in charge of performing the assessments were mainly people with access to the internals of the endeavors. It allowed for better access to the required information about the projects and developers, and it served at the same time to test the definition and documentation of metrics and indicators.

The selection of interviewees was guided by the way the company involved in the assessment was engaged with the F/OSS endeavor under scrutiny, rather than by quantitative objectives. For instance, in the cases of the assessment of the GCC back-end and Couverture, interviewees chosen were employees of AdaCore and at the same time active members of the related F/OSS community, because AdaCore is an integral and strategic component of this F/OSS community and the F/OSS community is to a considerable degree affected by the engagement of AdaCore. In the case of the Yanolc assessment, however, the

interviewees reflected exclusively the view of Océ, as there is no business at all involved in the Yanolc project, and the Océ perspective provided thus the only relevant business perspective. Finally, the Asterisk community is largely determined by the engagement of the US software company Digium and provides thus similarities to the GCC community. However, since in Europe the business perspective on Asterisk is rather provided by companies that implement Asterisk at their clients' sites than by code developers, we have selected interviewees from a service company (Freecode, a Norwegian company heavily involved in F/OSS). In other words: we made sure that the comparison of the measurement results to human perception and the assessment of user satisfaction from a business point of view complied with case specifics of these business perspectives for each F/OSS endeavor that was examined.

1.4 Strategy of WP5

T5.1 identifies the set of general hypotheses and a general protocol to follow in all our cases studies. The goal is that this general framework, made of hypotheses and general protocol, will steer our various hypotheses in a similar way so that their results can be aggregated at the end of WP5 to identify certain trends. Such an aggregation from several case study results is only possible if the various studies were performed in a compatible way. By providing a set of general hypotheses and a general protocol, T5.1 proposes a first action to help conduct our case studies in a compatible fashion.

T5.2 uses the general hypotheses and the general protocol created during T5.1, and refines specific hypotheses and a specific protocol for each case study. It is important that T5.2 controls how the refinement takes place because this is how compatibility across case studies will be guaranteed. Once the refinement has produced the specific hypotheses and specific protocol, including the specific questionnaires for a particular case, then T5.2 also has the responsibility to collect the data for that case study.

T5.3 first analyzes the data for each case study individually and assesses whether the specific hypotheses verify or not. Then, it aggregates their results to determine the usefulness and validity of the QualOSS methodology.

1.5 Structure of the Deliverable

The rest of the deliverable is structured as follows: Section 2 provides the definition of terms used in the document; Sections 3, 4, 5, and 6 provide the assessments of the different pilot projects, including their specific protocols, actual measurements, and interview results; Section 7 gives the overall conclusions for the performed assessments.

2. Terminology / Glossary

FIOSS Endeavor. F/OSS Endeavor is defined by 1) a set of work products, 2) the F/OSS community creating, updating and using these work products, 3) the tools used to act on these work products or to build or run the software product, and 4) the set of development processes executed by the community, these processes include rules and a division of labor accepted and followed by community members when interacting and creating work products.

3. AdaCore/GCC Version to Use as Back End for GNAT

3.1 Context

AdaCore uses the GNU Compiler Collection (GCC) back end for its flagship product, the GNAT compiler for Ada. Its target market is primarily interested in robustness and reliability, so quality metrics for the GCC back end related to its robustness are of paramount importance. AdaCore needs to upgrade to new versions of the GCC back end regularly (every one or two years) and the process to select the appropriate GCC version is critical for its business.

This assessment will analyze two different versions of the GCC back end (GCC 4.2 and 4.3) in the context of AdaCore's usage, and it will study whether the QualOSS methodology can help improving the decision process and increasing the confidence in the selected version.

3.2 Organization of the Assessment

The assessment method follows the assessment process prescribed by the QualOSS methodology. This means, that it conduct the following 5 tasks:

- Initiate the assessment
- Set up and plan the assessment
- Collect and analyze data
- Interpret the results of an assessment
- Supervise an assessment

The person in charge of doing most of the work of this assessment was José Ruiz, from AdaCore, taking advantage of his knowledge about GCC internals and its community and processes.

The people interviewed were Olivier Hainque (AdaCore) whose role is that of project manager of the GCC integration (and active contributor to GCC), and Olivier Ramonat (AdaCore), whose role is quality assurance at AdaCore.

3.3 GCC Back-End Assessment

The F/OSS use context of this assessment is that of an integration in a product (GCC back-end integrated as the code generator for the GNAT compiler), the F/OSS collaboration context is Full *F/OSS* Collaboration (AdaCore is part of the GCC community and they have collaborated in GCC development for a long time), the assessment mode is version comparison (version to integrate), and the F/OSS endeavor scope is for a part of a F/OSS project (only the GCC back-end part of the whole GCC project).

The version of the assessment used is the Standard QualOSS Assessment, Version 1.0_RC, which is a pre-release of version 1.0. The experience of using this candidate version was used to refine subsequent versions 1.0 and 1.1 of the Standard QualOSS Assessment.

3.3.1 Scoping the GCC Back-End Assessment

GCC Back-End 4.2 and 4.3	
Description:	*The GCC back-end code generator*
Official Website: *http://gcc.gnu.org*	
SCM Sites:	*svn://gcc.gnu.org/svn/gcc/branches/gcc-4_3-branch* *svn://gcc.gnu.org/svn/gcc/branches/gcc-4_2-branch*
Issue Tracking System: • http://gcc.gnu.org/bugzilla	

Mailing Lists:	• *http://gcc.gnu.org/lists.html*
	  • *http://gcc.gnu.org/ml/gcc (high volume list for general development discussions about GCC)*
	  • *http://gcc.gnu.org/ml/gcc-bugs (high volume list with mails from the Bugzilla bug-tracking system)*
	  • *http://gcc.gnu.org/ml/gcc-patches (relatively high volume list for patch submissions and discussion of particular patches)*

Forums: • N/A	

Package Distribution Sites :
• ftp://ftp.gnu.org/gnu/gcc
  • ftp://ftp.gnu.org/gnu/gcc/gcc-4.3.3
  • ftp://ftp.gnu.org/gnu/gcc/gcc-4.2.4	

Prog Lang.	*C, assembly*

The gcc-bugs mailing list uses markers in the subject to identify the different issues:

- [Bug Component/Number]

This is the information used for the specific queries. Component indicates the part of the technology to which it refers: targets, different languages supported, different phases in the compilation, etc (all these components are defined in http://gcc.gnu.org/bugzilla/describecomponents.cgi?product=gcc). For studying the GCC back-end, the relevant components to study are:

debug	A problem with generating debugging information or lacking debug information
inline-asm	A problem caused by use of inline assembly
lto	Problems relating to the Link Time Optimization (LTO) reading and writers
middle-end	GCC's middle end. Target dependent parts and optimization passes have their own component
rtl-optimization	A problem occurring in the Register Transfer Language (RTL) optimizers
target	Target specific issue
tree-optimization	A problem occurring in the tree-ssa (trees based on the Static Single Assignment) optimizers

The SVN repository contains subdirectories for the language-dependent part that must be ignored:

- Ada: gcc/ada, gnattools, libada
- Fortran: gcc/fortran, libgfortran
- Java: gcc/java, libjava, libffi, boehm-gc
- C++: gcc/cp, libstdc++-v3
- Objective C: gcc/objc, libobjc
- Objective C++: gcc/objcp

3.3.2 Work Products

GCC is one of the most important tools in the development of free software. GCC is free software, distributed under the GNU General Public License (GNU GPL), and its first release was in 1987. It runs on most platforms available today, and can produce output for many types of processors. It is very successful and expected to be pretty mature in terms of code, documentation, testing, and issue management. This section provides the results of the QualOSS assessment structured by the characteristics that the QualOSS Standard Assessment examines.

3.3.2.1 Maintainability

Metrics related to issue/enhancement management were very demanding in terms of resources (specific queries for the *Bugzilla* tracking system were developed), although it showed interesting results. The rapidity to implement bug fixes and enhancements was much better in GCC 4.3 than in 4.2 (roughly half of the time).

Metrics related to the code did not show a significant difference between versions 4.2 and 4.3. We needed to use tools like *cloc* (Count Lines of Code) and *SLOCCount* for counting code and comments, and *Pmccabe* for computing the complexity.

The time spent to get the metrics and characteristics related to product maintainability was around 6 hours for each assessment.

Characteristic	***GCC 4.2***	***GCC 4.3***
Percentage of unassigned issues	23.67%	24.28%
Average cyclomatic complexity per defined routine	8.61	11.95
Percentage of comments	19.92%	17.74%
Evolution of number of lines of code between successive releases	1.02%	3.90%
Average efferent coupling of low level modules	N/A	N/A
Average number of patch per issue	N/A	N/A
Evolution of change in code between minor releases	0.02%	0.47%
Evolution of cyclomatic complexity of defined routines between successive releases	0.03%	0.03%
Average efferent coupling of high level modules	N/A	N/A
Percentage of accepted enhancement proposals	37.07%	30.54%
Evolution of change to public interfaces between minor releases	0.02%	0.00%
Evolution of change in code between major releases	14.18%	17.63%
Rapidity of implementation of enhancement proposals	296	125
Rapidity of issue resolution	101	66
Evolution of change to public interfaces between major releases	11.16%	17.63%

3.3.2.2 Reliability

Metrics related to issue accounting and its evolution were very demanding in terms of resources (specific queries to the Bugzilla tracking system were developed), although it showed very interesting results. The number of serious regressions in GCC 4.3 was 30% smaller than in 4.2.

Metrics related to the change of code in a given branch showed also interesting results related to the location of changes. In GCC 4.3, 11.62% of the files changes from the first 4.3 release to the latest, while in the case of the GCC 4.2 branch the changes were much more pervasive (44.37% of the files changed).

We could not find tools for measuring code convention violations for C code, so these metrics and indicators are not available.

The time spent to get the metrics and characteristics related to product reliability was around 3 hours for each assessment.

Characteristic	***GCC 4.2***	***GCC 4.3***
Stability evolution	347 regression 216 serious regressions 5 minor releases to achieve 20% less regressions than previous one	284 regressions 151 serious regressions 1 minor releases to achieve 20% less regressions than previous one
Stability evolution specific version	347 regressions 216 serious regressions Decreasing number of bugs in major releases	284 regressions 151 serious regressions Decreasing number of bugs in major releases
Importance of corrections	44.37% files changed between 4.2.0 and 4.2.4	11.62% files changed between 4.3.0 and 4.3.3
Coding convention violations	N/A for C	N/A for C

3.3.2.3 Security

There were just a few occurrences of vulnerabilities related to the GCC compiler so there is not much relevant information that can be extracted (this is not the type of software that is subject to this kind of issues).

The time spent to get the metrics and characteristics related to product reliability was around 2 hours for each assessment.

Characteristic	***GCC 4.2***	***GCC 4.3***
Global track record of all NVD entries over time	5 occurrences	5 occurrences
Global track record of High Severity NVD Entries over time	2 occurrences	2 occurrences
Global track record of Medium Severity NVD Entries over time	1 occurrence	1 occurrence
Predictability of yearly trend of NVD Entries over major releases	Low number and low variation	Low number and low variation
NVD Entry Status of selected release	1 entry recent	1 entry recent
High Severity NVD Entry Status of selected release	0 entries recent	1 entry recent
Track record of NVD Entries for selected minor over time	Very low number of entries	Very low number of entries
Track record of High Severity NVD Entries for selected minor over time	Very low number of entries	Very low number of entries

3.3.2.4 Documentation

The documentation is very stable within the GCC endeavor, and therefore the QualOSS assessment could not find any difference between GCC 4.2 and 4.3. In terms of type of documentation available GCC showed a pretty high score (more than 75%), but the contents of these documents are weaker (38% of the expected contents).

There is a pretty extensive list of documents to look at, and this is the reason why the assessment of the documentation took around 1 day of work.

Characteristic	*GCC 4.2*	*GCC 4.3*
Document type availability	76.72%	76.72%
Documentation information availability	37.97%	37.97%

3.3.2.5 Tests

The testing procedures are very stable for a mature endeavor like GCC, and therefore the QualOSS assessment could not find big differences between GCC 4.2 and 4.3. GCC relies on extensive black-box testing, which is very common in this kind of big and complex tools.

The assessment of the testing part took around 6 hours, and the most time-consuming part was the code coverage analysis. The gcov tool was used for coverage.

Characteristic	*GCC 4.2*	*GCC 4.3*
Likelyhood future test reports	Test reports for all minor releases	Test reports for all minor releases
Test report availability	Only system tests	Only system tests
Environment test availability	Only system tests	Only system tests
Unit test coverage adequacy	No unit test	No unit test
System test coverage adequacy	78.14%line 78.80% function 97.73% file	77.42%line 77.47% function 97.30% file
Unit test suite adequacy	No unit test	No unit test
System test suite adequacy	14 systems tested and well covered	26 systems tested and well covered
Ease of testing	Documented procedure	Documented procedure

3.3.3 Community Members

There is a very big community of users/maintainers behind GCC with hundreds of different contributors and tens of people maintaining and contributing to GCC on a daily basis. Most contributors work for big companies with vested interests in making GCC work in their environment (RedHat, Google, IBM, AMD, Intel, etc.). Given the nature of a development community and the age and stability of the GCC contributing community, we measured these characteristics for the whole GCC instead of the concrete versions. The results showed a pretty stable community and with a reasonable distribution of the workload among community members.

The assessment of these software processes took around 6 hours.

Characteristic	*GCC*
Evolution of new contributors (code)	Slightly increasing
Evolution of new contributors (other than code)	Slightly increasing
Evolution of new core contributors	Stable the last 10 years
Core members stopping contributing	Slightly increasing
Core members number	Decreasing

Longevity of commiters	24.5 months
Number of events	Stable
Territoriality	55.50%
Lines per committer	23419.4
Percentage of handled files	15.60%

3.3.4 Software Processes

Software processes are exactly the same for the two GCC versions, as would be expected in a mature endeavor. Most of the processes are well documented and followed, but there are some which are not documented and that form part of the common, but not formalized, community knowledge.

The assessment of these software processes took around 4 hours.

Characteristic	*GCC 4.2*	*GCC 4.3*
Change submission maturity	Documented and followed	Documented and followed
Change review maturity	Undocumented but mature	Undocumented but mature
Change review adequacy	Review required	Review required
Committer promotion maturity	Undocumented (sponsorship)	Undocumented (sponsorship)
Commit review maturity	Undocumented	Undocumented
Commit review adequacy	Bug tracking and version control linked	Bug tracking and version control linked
Enhancement proposal maturity	Documented and followed	Documented and followed
Enhancement proposal adequacy	Justified answers	Justified answers
Issue management maturity	Documented and followed	Documented and followed
Issue management adequacy	Patch handling documented, no CVE link	Patch handling documented, no CVE link
Testing process maturity	Documented and followed	Documented and followed
Testing process adequacy	Test suite and validation documented	Test suite and validation documented
Release planning maturity	Documented and followed	Documented and followed
Release planning adequacy	Clear definition of stages for patch acceptance	Clear definition of stages for patch acceptance
Release management maturity	Documented and followed	Documented and followed
Release management adequacy	Well defined procedure for branching and packaging	Well defined procedure for branching and packaging
Release backport maturity	Undocumented	Undocumented

3.4 Human Perception

The people interviewed were:

- Olivier Hainque (AdaCore) whose role is that of project manager of the GCC integration (and active contributor to GCC)
- Olivier Ramonat (AdaCore), whose role is quality assurance at AdaCore

3.4.1 Perception of GCC back-end before the QualOSS assessment

The pre-assessment perception of the quality of the GCC back-end was characterized by two determinants that were mentioned by both interviewees: 1) the expectation that a quality measurement should mainly, if not exclusively, be based on metrics, i.e. numeric data, focusing on product features and software processes and, 2) the observation that there are significant differences between the two GCC versions that were compared, implying that these differences must be reflected in the measurement results in order to appear trustworthy and reliable.

We must point out that the two interviewees from AdaCore were asked to consider the two GCC back-end versions from a business point of view, as this is the perspective from which the QualOSS assessment is performed. However, we also have to take into account that both interviewees were not only AdaCore employees but also GCC community members. Though AdaCore membership and GCC community affiliation coincide to some degree, as AdaCore considers itself as “the GNAT Pro Company” and thus as part of a F/OSS community, it is evident that at least partially the life in the community follows other principles and values than the commercial aspect that dominates a company's perception. This was well illustrated in the interviews by both interviewees assuming a “double we” perspective: we at AdaCore, we at the GCC community.

In this double role, the two AdaCore interviewees must be located somewhere in between the company and the community. As company members, they both worried about the usability of GCC back-end (especially with regard to version 4.2), as community members they both were proud of the level of quality their product has achieved. Like pure community members but too a much lesser degree than observed in the perception of the QualOSS assessment from the community point of view (see D3.3), both interviewees showed sometimes an “introspective self-sufficiency”, i.e. they were pleased if something was good or functioning from the community point of view and did not bother much about the possible requirements of other users.

The discussion with the interviewees on the details of the QualOSS Standard Assessement indicated that GCC users and stakeholders seem to focus on specific aspects of “their” F/OSS endeavor, either those issues they personally are most concerned about, such as the quality of new code, or those parts on which they personally are mainly involved in. As a result, in the beginning none of the interviewees saw much sense in the provision of community metrics, but later in the discussion they changed their mind and argued that the community measures of the QualOSS standard assessment provide a valid core information when risks to deal with this endeavor are assessed from a business point of view.

In particular, the pre-assessment of the GCC back-end quality showed that:

- at the level of work products:
  - code quality was assessed quite positively (“no risk from a business point of view” was assigned especially to reliability and security)
  - documentation was considered to be of good quality, though some uncertainty regarding structuredness and completeness could be observed, so that for these two items a risk assessment was not possible
  - tests were considered to be provided for the overall GCC software code, therefore a specific assessment of tests regarding the GCC back-end was difficult to do
- all the community measures were assessed positively with regards to risks from a business perspective
- software process aspects were assessed quite heterogeneously, as:
  - change management capability and support management capability were considered to be of very good or good quality, thus providing no or only a limited risk

- requirements management capability and community management capability were considered to provide no or some risk
- release management capability was considered to provide quite a risk

3.4.2 Perception of GCC back-end after the QualOSS assessment

The post measurement perception of the measurement results of the two interviewees diverged considerably. While Olivier Ramonat expressed his surprise about some of the results at the level of detailed measures and said that he considers some of these as wrong, Olivier Hainque's perception was that the results confirm very well the firm's own impression of the quality of the two GCC back-end versions. However, while Olivier Ramonat confirmed that overall the outcome of the measurement appears reliable to him, Olivier Hainque also found results that contradicted his own impression. For instance, while the QualOSS measurement indicates that the cyclomatic complexity of the GCC back-end has increased he was assuming that it was decreasing. Thus, as a conclusion, both interviewees found results they did not expect but overall they perceived the QualOSS Standard Assessment as reliable.

Both interviewees perceived the huge number of measures as positive but also pointed out that some of these measures might not be easy to interpret with regard to their impact on quality. For instance, while it was perceived very positively to have a measure indicating the celerity at which code enhancements are made, it was criticized that it is not possible to distinguish important from unimportant enhancements (e.g. with regard to their effects on functionality).

The comparison of the measurement results and the perception of the quality of GCC back-end by the two interviewees in more detail shows that, by and large, the human perception corresponds pretty much to the overall measurement. The human perception appears a bit more positive (i.e. indicating less risk from a business point of view) than the QualOSS results, but these differences can be neglected, as usually the interviewees tended towards assigning a “green” where the QualOSS Standard Assessment turned out a “yellow”, while larger discrepancies do not appear. Security and test aspects provide the only measures that were perceived more risky (“yellow”) by the interviewees than by the QualOSS Standard Assessment.

4. Freecode Asterisk

4.1 Context

Freecode uses Asterisk for the provision and implementation of complete telecommunication infrastructures. This case study investigates how Freecode performs within the Asterisk endeavor in order to secure and improve the quality of its products and services. Asterisk has not yet succeeded in convincing the Asterisk community to incorporate its patches, so that Freecode now owns a set of patches that, if implemented, would ease Freecode's business because they expect better community support when these changes would be implemented in the trunk version of Asterisk. However, Freecode highly appreciates community support, and the company prefers to implement the public available trunk version of Asterisk instead of their own version which does not receive support from the community. Asterisk wants to know if collaboration with the community can be improved and what could be done in order to get their patches incorporated in the trunk version of Asterisk. The latter would save the company both effort and money when implementing Asterisk.

Freecode would like to compare QualOSS results on Asterisk 1.4.26 against an internal OpenBRR assessment that they performed on the same version.

4.2 Organization of the Assessment

The assessment method follows the assessment process prescribed by the QualOSS methodology. This means, that it conduct the following 5 tasks:

- Initiate the assessment
- Set up and plan the assessment

- Collect and analyze data
- Interpret the results of an assessment
- Supervise an assessment

The people in charge of doing this assessment were Ruediger Glott for understanding Freecode's needs and scoping, and Jean-Christophe Deprez and José Ruiz for the data collection.

The people interviewed were Tommy Jensen (Freecode), Anna Tannenberg (Freecode), and Arne-Kristian Groven (Norsk Regnesentral, Oslo), who are very knowledgeable about quality assessments and Asterisk.

4.3 Asterisk Assessment

The F/OSS use context of this assessment is that of a service integration (Freecode uses Asterisk to provide telecommunication services), the F/OSS collaboration context is Full *F/OSS* Collaboration (Freecode is part of the Asterisk community and they have collaborated in Asterisk for a long time), the assessment mode is instrospection (how to better collaborate with the Asterisk community), and the F/OSS endeavor scope is for a part of a single F/OSS project (Asterisk). The version of the assessment used is the Standard QualOSS Assessment, Version 1.1.

4.3.1 Scoping the Asterisk Assessment

Asterisk 1.4.26	
Description:	*Asterisk is the world's leading open source Private Branch eXchange (PBX) telephony engine and telephony applications toolkit*
Official Website: *http://www.asterisk.org*	
SCM Sites:	*http://svn.asterisk.org/svn/asterisk/tags/1.4.26*
Issue Tracking System: • http://issues.asterisk.org	

Mailing Lists:	• *http://lists.digium.com/mailman/listinfo*
	  • *http://lists.digium.com/mailman/listinfo/asterisk-bugs*
	  • *http://lists.digium.com/mailman/listinfo/asterisk-commits*

Forums: • N/A	
Package Distribution Sites : • *http://www.asterisk.org/downloads*	
Prog Lang.	*C*

For measurements where we need some history (to see the evolution) we looked at versions 0.1.0, 1.0.0, 1.2.0, and 1.4.0, in addition to the 1.4.26. For community, the complete 1.4 branch was considered.

4.3.2 Work Products

Asterisk is the world's most popular open source telephony project. Under development since 1999, Asterisk is free, open source software, originally written by Mark Spencer of Digium, Inc., which has been contributed from open source software engineers around the world. Currently boasting over two million users. It is very successful and it is expected to be pretty mature in terms of code, documentation, testing, and issue management. This section provides the results of the QualOSS assessment grouped by characteristic.

4.3.2.1 Maintainability

Metrics related to issue/enhancement management could not be extracted reliably from the bug tracker. The system used, *Mantis*, does not support the extraction of the required information in an automatic way. Therefore, issue/enhancement information was extracted manually, but it took a reasonable amount of time because there were not many instances to look at manually. It shows a code which changes extensively, which is rather complex and which features a relatively low amount of comments. However, coupling is low, and issues are fixed very quickly.

The time spent to get the metrics and characteristics related to product maintainability was around 3 hours.

Characteristic	***Asterisk 1.4.26***
Percentage of unassigned issues	67.74%
Average cyclomatic complexity per defined routine	8.4
Percentage of comments	1.74 line of comment / complexity
Evolution of number of lines of code between successive releases	Rapidly increasing
Average efferent coupling of low level modules	0.54
Average number of patch per issue	N/A
Evolution of change in code between minor releases	34.07%
Evolution of cyclomatic complexity of defined routines between successive releases	0.16%
Average efferent coupling of high level modules	1.41
Percentage of accepted enhancement proposals	0
Evolution of change to public interfaces between minor releases	40.95%
Evolution of change in code between major releases	88.42%
Rapidity of implementation of enhancement proposals	None
Rapidity of issue resolution	8.93
Evolution of change to public interfaces between major releases	85.99%

4.3.2.2 Reliability

Metrics related to issue accounting and its evolution were problematic, because the bug tracker (*Mantis*) does not contain information for versions older than 1.4. Therefore, the evolution needed to be addressed by dates corresponding to the day of the release. For the evolution of minor releases (1.4.26, 1.4.26.1, 1.4.26.2, 1.4.26.3) this did not make sense so they were not measured. The amount of issues in Asterisk is very high.

We could not find tools for measuring code convention violations for C code, so these metrics and indicators are not available.

The time spent to get the metrics and characteristics related to product reliability was around 3 hours.

Characteristic	***Asterisk 1.4.26***
Stability evolution	Average of 2138 issues / year
Stability evolution specific version	N/A
Importance of corrections	In the serie 1.4.26, 1.4.26.1, 1.4.26.2, and 1.4.26.3: 35 files changed on average

	2.44% changed
Coding convention violations	N/A for C

4.3.2.3 Security

Asterisk belongs to the kind of software that may be sensitive to security threats. Measurements of vulnerabilities until the end of 2008 were computed automatically (downloading the NVD XML file), and those for the 1.4.26 version (which was released in July 2009) were searched using the interactive search tool. There are not too many occurrences of security issues, but the mayor problem is that the number of high-severity ones is relatively high and increasing over time.

The time spent to get the metrics and characteristics related to product reliability was around 3 hours 20 minutes.

Characteristic	***Asterisk 1.4.26***
Global track record of all NVD entries over time	6.83 occurrences / year
Global track record of High Severity NVD Entries over time	4 occurrences / year (increasing)
Global track record of Medium Severity NVD Entries over time	2.67 occurrences / year
Predictability of yearly trend of NVD Entries over major releases	Increasing and high variability
Predictability of yearly trend of High Severity NVD Entries over major releases	Increasing and high variability
NVD Entry Status of selected release	1 entry for 1.4.26
High Severity NVD Entry Status of selected release	No entries for 1.4.26
Track record of NVD Entries for selected minor over time	9.33 entries (average per minor)
Track record of High Severity NVD Entries for selected minor over time	5.67 entries (average per minor)

4.3.2.4 Documentation

Many of the documentation types are present (the more prominent exceptions are the requirement, design and maintenance documents) but the level of detail and formalism in the documentation is low. There is a big amount of knowledge which is in mailing lists and the developers themselves.

There is a pretty extensive list of documents to look at, and the assessment of the documentation took around 5 hours.

Characteristic	***Asterisk 1.4.26***
Document type availability	76.92%
Document type availability (weighted)	75.00%
Document information availability	19.94%
Document information availability (weighted)	20.86%
Document information availability (weighted organisation and completeness)	20.74%
Document information availability (weighted document type and organisation and completeness)	21.69%

4.3.2.5 Tests

The testing procedures are not formally defined. Asterisk relies extensively on human test effort, with few automated testing, and it expect its community to act as testers.

The assessment of the testing part took around 1 hour and 30 minutes.

Characteristic	*Asterisk 1.4.26*
Likelyhood future test reports	No test reports
Test report availability	No test reports
Environment test availability	No test suites
Unit test coverage adequacy	No unit test
System test coverage adequacy	No system test
Unit test suite adequacy	No unit test
System test suite adequacy	No system test
Ease of testing	No automatic testing

4.3.3 Community Members

There is a big community of users/maintainers for Asterisk, which is more or less stable, but with people doing very local changes.

The assessment of these software processes took around 3 hours and 10 minutes.

Characteristic	*Asterisk 1.4.26*
Evolution of new community members reporting bugs	N/A
Evolution of new community members contributing code	Stable
Evolution of new community members contributing other than code	Stable
Evolution of new core contributors	Increasing
Core members stopping contributing	Increasing
Core members number	Increasing
Longevity of commiters	14.89 months
Evolution of people contributing patches/changes	Increasing
Number of events	N/A
Evolution of number of commits	Slightly decreasing
Territoriality	94.48%
Lines per committer	26
Percentage of handled files	14.89%

4.3.4 Software Processes

Asterisk is a mature endeavor, and software processes are rather well established, except for the testing (which relies a lot on manual testing).

The assessment of these software processes took around 2 hours and 30 minutes.

Characteristic	*Asterisk 1.4.26*
Change submission maturity	Documented and followed
Change review maturity	Undocumented but mature
Change review adequacy	Review and testing required
Committer promotion maturity	Undocumented
Commit review maturity	Undocumented but mature
Commit review adequacy	Review and testing required, link between bug tracker and CM
Enhancement proposal maturity	Documented and followed
Enhancement proposal adequacy	Justified answers
Issue management maturity	Documented and followed
Issue management adequacy	Documented, traceable to bug tracker and CVE
Testing process maturity	Undocumented
Testing process adequacy	No test suite or validation procedure
Release planning maturity	Documented and followed
Release planning adequacy	No dates
Release management maturity	Undocumented
Release management adequacy	Undefined procedure for branching and packaging
Release backport maturity	Undocumented

4.4 Human Perception

People interviewed:

- Tommy Jensen (Freecode), with the role of quality assurance and Asterisk expert
- Anna Tannenberg (Freecode), with the role of quality assurance and Asterisk expert
- Arne-Kristian Groven (Norsk Regnesentral, Oslo), with the role of quality assurance

4.4.1 Perception of Asterisk 1.4.26 before the QualOSS assessment

The human perception of the quality of Asterisk 1.4.26 was examined by interviewing two representatives of a Norwegian company (Freecode) providing services based on Asterisk. In addition, synergy effects with the EUX2010Sec project (www.eux2010sec.nr.no) have been tapped in order to get feedback from a measurement expert from the Norsk Regnesentral in Oslo. Finally, the collaboration with EUX2010Sec provided an opportunity to directly compare the QualOSS Standard Assessment to OpenBRR, as Norsk Regnesentral recently carried out an OpenBRR analysis of Asterisk 1.4.26. However, given the purpose of the QualOSS project, and in order to keep the results on the different FLOSS endeavors presented in this document comparable, we will keep digresses to the OpenBRR results to a minimum.

The pre-assessment perception of Asterisk by the Freecode representatives was characterized by deep knowledge of the product. Overall, Asterisk was considered to be of high quality. Problems were mainly seen with regard to the implementation of enhancements in the trunk version of Asterisk, as Freecode occasionally encountered difficulties to get patches they have developed accepted by the community led by Digium. The reasons behind these difficulties were seen rather in cultural differences than in quality aspects.

Freecode considers itself as *the* Norwegian open source company and has internalized and emphasized very much F/OSS principles, while Digium and the rest of the community seem to follow an approach that deviates to some extent from pure F/OSS principles.

In particular, the pre-assessment of the quality of Asterisk 1.4.26 showed that all work product related matters were considered to be of high quality, providing no risk for business use. The same applied to software processes, with the reservation regarding the communication issues depicted above. The community measures, however, provided a point of concern for Freecode as the company faced difficulties to understand the way the community works in detail and to interact with the community so that it would be easier for Freecode to get their changes implemented. However, with regard to risks, as perceived from a business point of view, none of the points that concerned Freecode were regarded as something that provided such a risk. In other words, the company showed a very positive attitude towards Asterisk.

4.4.2 Perception of Asterisk 1.4.26 after the QualOSS assessment

Freecode has experience in quality assessments, as it constantly observes how Asterisk develops, and Freecode has also analyzed Asterisk with OpenBRR. Therefore, the perception of the results of the QualOSS Standard Assessment of Asterisk 1.4.26 was characterized by the fact that these results were compared not only to the interviewee's individual impression of the quality of Asterisk, but also to the formalized knowledge about Asterisk's quality the company has established.

In this perspective, the QualOSS assessment was perceived to be better than OpenBRR because its measures appeared overall better defined, and thus less prone to ambiguities than some measures of OpenBRR.[1] However, it was criticized that the QualOSS Standard Assessment appears too detailed and covers too many measures, so that it was hard to get an overview of the total picture. However, it must be said that at the time of the interview the results of the measurement could only be retrieved from spreadsheets that were indeed hard to read. With the on-line visualization of the results that is meanwhile available this criticism should have become obsolete.

Another point that was criticized by the interviewees was that, from the perspective of a service company that implements Asterisk at the sites of its clients, many QualOSS measures appeared too technical and are therefore not usable as a “marketing argument” because the end user (i.e. Freecode's clients) would not be able to understand them. However, since the QualOSS Assessment has never been designed for the needs of end users, we do not think that this point, though it is true of course, abates the usability of the QualOSS Standard Assessment. Nevertheless, the interviewees confirmed that a company like Freecode has no difficulties to understand the measurement as long as the measurement methodology for each measure and indicator is well documented.

Further, a point that was discussed by the interviewees was the problem of how to interpret the combined effects of different measures on the value of the indicator composed of these measures. One of the examples in this context was provided by the measures of code change and how they affect maintainability. However, at the end of all these discussions, it turned out that the indicators and their compositions appear thought through very well.

Overall, for firms, the QualOSS Standard Assessment was judged better than OpenBRR results, but appeared too detailed and too technical to be understood by end users. Therefore, for firms' clients, OpenBRR appeared to be more utile than the QualOSS Standard Assessment. On the other hand, the richness of the information provided by QualOSS was seen an an advantage especially for a firm with deep knowledge of the F/OSS endeavor and the knowledge, resources and interest in monitor the advancement of this endeavor over time. Moreover, the number of measures and their documentation were considered as an advantage with regard to the repeatability and verifiability of the measurement results. Taking all points together, the interviewees had the impression that the QualOSS Standard Assessment provides a reliable insight in the quality of a F/OSS endeavor.

[1] One questionable measures of OpenBRR, in this regard, is, for instance, "adoption", which is measured by the number of books containing the component name that is retrievable from a PowerSearch query at amazon.com.

5. Océ PRISMAspool to use the LPR client of the yanolc project

5.1 Context

Océ delivers the Océ PRISMAspool software product that provides an enhanced spooling system for Windows environments. PRISMAspool uses the Line Printer Remote (LPR) protocol in 2 use cases. The implementation currently uses the LPR client delivered with the Windows operating systems. A new LPR client should be delivered with a next version of PRISMAspool in order to solve some restrictions. This case study will analyze the adequacy of using a subset of the Open Source project *yanolc* (Yet ANOther Lpr Client) as a basis for this delivery. As PRISMAspool is used for professional printing, for transactional and mailing environment, the requirements regarding quality and reliability are extremely important.

This assessment analyzes the most recent available version of yanolc, namely version 1.2.11, and it will study whether the QualOSS methodology will confirm the decision that has been taken to not use yanolc as a basis but well an internally developed tool.

5.2 Organization of the Assessment

The assessment method follows the assessment process prescribed by the QualOSS methodology. This means, that it conduct the following 5 tasks:

- Initiate the assessment
- Set up and plan the assessment
- Collect and analyze data
- Interpret the results of an assessment
- Supervise an assessment

The person in charge of doing most of the work of this assessment was Jacques Flamand, working part time in Océ, so he could use his knowledge about yanolc internals and its community and processes.

The people interviewed were Jacques Flamand (Océ), whose role is that of project and product manager of PRISMAspool, and François Thiry (Océ), who is one of the developers of PRISMAspool.

5.3 yanolc Assessment

The F/OSS use context of this assessment is the integration in a product (a subset of yanolc delivered as an additional tool with PRISMAspool), the F/OSS collaboration context is fork (mainly due to the fact that only a subset of the functionality should be provided), the assessment mode is the comparison of using yanolc vs. developing an own tool, and the F/OSS endeavor scope is the whole F/OSS project.

The version of the assessment used is the Standard QualOSS Assessment, Version 1.0_RC, which is a pre-release of version 1.0. The experience of using this candidate version was used to refine subsequent versions 1.0 and 1.1 of the Standard QualOSS Assessment.

5.3.1 Scoping the yanolc Assessment

YANOLC	
Description:	*yanolc*
Official Website: *http://yanolc.sourceforge.net/*	
SCM Sites:	*Not available* *(only packaged source are available)*
Issue Tracking System:	

• http://sourceforge.net/tracker/?group_id=96215&atid=614047	
Mailing Lists:	*N/A*
Forums: • *http://sourceforge.net/projects/yanolc/forums*	
Package Distribution Sites : • http://sourceforge.net/project/showfiles.php?group_id=96215&package_id=102739	
Prog Lang.:	*C*

The scope is defined as follows :

Target version : 1.2.11

Additional versions to be considered as correction version : 1.2.10, 1.2.9, 1.2.8, 1.2.7

5.3.2 Work Products

5.3.2.1 Maintainability

Measures have been strongly influenced by the fact that no information could be found in any bug tracker. The assessment of maintainability needed around 4 hours, and the most time-consuming part was the code analysis.

The code has a rather high complexity, but it does not change much.

Characteristic	***yanolc 1.2.11***
Percentage of accepted enhancement proposals	No information on enhancement proposal
Rapidity of implementation of enhancement proposals	No information on enhancement proposal
Evolution of change in code between major releases	0.86
Evolution of change to public interfaces between major releases	0
Evolution of number of lines of code between successive releases	0.09
Percentage of unassigned issues	No information on issues
Rapidity of issue resolution	No information on issues
Evolution of change in code between minor releases	0.03
Average number of patch per issue	No information on patches
Evolution of change to public interfaces between minor releases	0
Percentage of high level modules involved in dependency cycles	0
Average efferent coupling per low level module	0
Average cyclomatic complexity per defined routine	42.6

Percentage of commented algorithm	0.11
Evolution of cyclomatic complexity of defined routines between successive releases	-0.01

5.3.2.2 Reliability

Again, measures have been strongly influenced by the fact that no information could be found in any bug tracker.

No tool was available for measuring code convention violations for C code, so these metrics and indicators are not available.

The assessment of Maintainability needed around 1,5 hours.

Characteristic	***yanolc 1.2.11***
Stability_Evolution	0
Stability_Evolution_Specific_Version	0
Importance_of_Corrections	2.5
coding_convention_violation	n.a.

5.3.2.3 Security

There were no occurrence of vulnerabilities related to yanolc so there is not much relevant information that can be extracted.

The time spent to get the metrics and characteristics related to product security was around 0,5 hour.

Characteristic	***yanolc 1.2.11***
Global track record of all NVD entries over time	No occurence
Global track record of High Severity NVD Entries over time	No occurence
Global track record of Medium Severity NVD Entries over time	No occurence
Predictability of yearly trend of NVD Entries over major releases	No occurence
NVD Entry Status of selected release	No occurence
High Severity NVD Entry Status of selected release	No occurence
Track record of NVD Entries for selected minor over time	No occurence
Track record of High Severity NVD Entries for selected minor over time	No occurence

5.3.2.4 Documentation

Relatively few documents have been found for this project; the assessment of the documentation needed around 2 hours.

Characteristic	***yanolc 1.2.11***
Document type availability	46.15%
Documentation information availability	18.41%

5.3.2.5 Tests

No test material has been found for this project; the assessment of the testing part took around 1 hour.

Characteristic	*yanolc 1.2.11*
Likelyhood future test reports	No report
Test report availability	No report
Environment test availability	No test material
Unit test coverage adequacy	No test material
System test coverage adequacy	No test material
Unit test suite adequacy	No test material
System test suite adequacy	No test material
Ease of testing	No test material

5.3.3 Community Members

No measurement could be introduced in the spreadsheet because the used data sources (source repository, bug tracker) are not available for this project. Based on the evidence-based hypothesis that one single developer worked on the yanolc project, indicators have been estimated and manually introduced in the spreadsheet.

The assessment of this characteristic needed about 1 hour.

Characteristic	*yanolc 1.2.11*
Evolution first bug submitted by registered people	One single developer
Evolution first commit submitted by registered people	One single developer
Evolution first no source code commit by registered people + evolution first post in mailing lists + evolution first no bug report submitted by registered people	One single developer
Evolution new core contributors	One single developer
Evolution core member leaving core team	One single developer
Evolution balance core team	One single developer
Average committers longevity	One single developer
Evolution code contributors who submitted patches and changes	One single developer
Total code contributors who submitted patches and changes	One single developer
Evolution number of events	One single developer
Evolution number of commits	One single developer
Percentage people working old releases	No information
Territoriality	No information
Lines per committer + bugs per commiter + emails per committer	No information
Lines per committer	10714 locs
Percentage number handled files	11 files

5.3.4 Software Processes

No documented process has been detected on the project; measurements and indicators have been introduced manually in the spreadsheet.

The assessment of software processes took around 1,5 hours.

Characteristic	*yanolc 1.2.11*
Change submission maturity	Undocumented
Change review maturity	Undocumented
Change review adequacy	Undocumented
Committer promotion maturity	Undocumented
Commit review maturity	Undocumented
Commit review adequacy	Undocumented
Enhancement proposal maturity	Undocumented
Enhancement proposal adequacy	Undocumented
Issue management maturity	Undocumented
Issue management adequacy	Undocumented
Testing process maturity	Undocumented
Testing process adequacy	Undocumented
Release planning maturity	Undocumented
Release planning adequacy	Feature-based releases
Release management maturity	Undocumented
Release management adequacy	Releases seem match objectives
Release backport maturity	Undocumented

5.4 Human Perception

People interviewed

- Jacques Flamand (Océ), project and product manager of PRISMAspool
- François Thiry (Océ), one of the developers of PRISMAspool

5.4.1 Perception of Yanolc 1.2.11 before the QualOSS assessment

The expectations of the two Océ representatives from yanolc 1.2.11 as compared to the PRISMAspool LPR client were very clear: they wanted a product that provides superior quality especially with regard to code, documentation, and community support.

When Océ began to examine yanolc 1.2.11, the people assigned to this task found out that in all respects the internal resources available for handling and maintaining the PRISMAspool LPR client were superior to yanolc 1.2.11. This applied especially to code quality and software process factors. In addition, the examination of yanolc 1.2.11 also suggested that the support the Océ staff can provide for the PRISMAspool LPR client was much better than the support that could be expected from the F/OSS community for yanolc 1.2.11 because there was no real community behind yanolc, as its development and maintenance relied on only one person.

Especially the latter point was considered to result in high risks Océ would encounter when using yanolc 1.2.11 without making sure that it can be implemented and maintained with internal resources. Therefore,

and because Océ needed only a part of yanolc's functionality, the company decided that forking yanolc would be the best option. Both interviewees showed some discontent and frustration about this, as community support was one of the big advantages they hoped to receive from using a F/OSS product. According to the opinion of both interviewees, in the case of very small F/OSS communities or communities that deem for other reasons to be unable to secure maintenance and support, a fork is the best option for a company.

This condition has a strong impact on the human perception of the quality of yanolc through the Océ representatives that we have interviewed. In contrast to other F/OSS collaboration contexts, a fork means that the company relies rather on its own capacities to modify, implement and maintain the program (or parts of it) it needs than on the community around this program. Without implying any negative notion this term may have, the case of a fork collaboration context is thus somewhat “egocentric”: It does not matter what others can contribute, it only matters if the company can handle the required tasks with its own resources.

However, both interviewees pointed out that the risks of a fork would increase dramatically if a company overestimates its own capabilities in handling the desired F/OSS program. Therefore, a fork provides a F/OSS collaboration context that requires from firms not only to assess carefully what the demanded F/OSS product and its community can contribute to its purposes, but also to check equally carefully what knowledge, skills and resources the company has available in order to perform those tasks internally, which otherwise would be expected from the F/OSS community.

5.4.2 Perception of Yanolc 1.2.11 after the QualOSS assessment

Overall, the assessment results were considered to be reliable, i.e. to show a correct picture of the risks associated with yanolc 1.2.11 when considered from a business perspective. The huge number of measures was considered to provide a very comprehensive overview and deep insight in yanolc 1.2.11 and thus a big advantage over the measurability of proprietary software products.

It was highlighted that though “black” is the prevailing color this does not automatically imply that yanolc poses a high risk for businesses in any case. Jacques Flamand emphasized that the context in which yanolc is considered to be used by a company is decisive. If a company would like to rely on external support and maintenance for yanolc, the large share of “black” in the assessment results indeed indicates high risks. However, in case of a fork, as considered by Océ, the high risk that was assigned to some items could be countered by Océ's own expertise and resources.

In other words: The perception of the results of the QualOSS Standard Assessment by the two interviewees was highly determined by the fact that due to the decision to fork yanolc 1.2.11 many aspects had to be considered from the perspective of Océ's resources and capabilities. For instance, while community support and software processes were considered to be crucial in the pre-assessment interviews, these were now considered as quite irrelevant because the interviewees knew that they can rely on the expertise of Océ staff.

Another point that was discussed after the QualOSS Standard Assessment was the way how the “high risk” results have been generated by the assessment methodology. In some cases, black was assigned because no data was available. In other cases, black was assigned because yanolc 1.2.11 did not fully comply with F/OSS principles. However, in order to interpret these results in a meaningful way, the user of the results should be able to understand some details of the measurement and that the meanings of the results may vary in different F/OSS use contexts. For instance, while maintainability of yanolc 1.2.11 was assigned a high risk for businesses, for Océ yanolc can be maintained quite easily. Thus, a large share of results that indicate a high risk for businesses should not automatically imply a “hands off” signal to businesses but encourage them to think about using the F/OSS product they are interested in in a different collaboration context, say, in form of a fork.

Regarding the individual risk assessment of the two interviewees as compared to the risk assessment performed by the QualOSS Standard Assessment, the two interviewees showed strong differences in their overall assessment but some similarities in the assessment of the separate items. Though both interviewees

commonly say that, to them, the QualOSS Standard Assessment indicates that yanolc 1.2.11 is of medium quality, Jacques Flamand assigned the overall quality of yanolc 1.2.11 a “high risk” (red), while François Thiry assigned it “some risk” (yellow).

A more detailed view at the individual perception of yanolc's quality after the results of the QualOSS Standard Assessment have been showed to the interviewees illustrates that:

- Both interviewees agree that the work product related factors are by and large not risky or risky only to a limited extent, thus being more positive than the QualOSS Standard Assessment suggests. The reason for this discrepancy is the focus on Océ's own capacities to handle problems in this field.
- The documentation related factors are also equally assessed by both interviewees, as they both say that documentation is posing quite some risk or a high risk to businesses. In this regard, the interviewees' perception is in line with the results of the QualOSS Standard Assessment.
- The test related factors are commonly assessed as posing quite some risk or a high risk to businesses, which is in line with the results of the QualOSS Standard Assessment.
- The community related factors show a strong discrepancy between the individual perceptions of the two interviewees. While Jacques Flamand's perception goes well in line with the result of the QualOSS Standard Assessment, which attributes a high risk to yanolc 1.2.11 in this regard, François Thiry considers the community related quality factors as acceptable and thus deviates considerably from the QualOSS assessment results. The reason for this discrepancy seems to be twofold: on the one hand, François Thiry showed a lot of respect towards the effort and performance of the person who created yanolc single-handedly, on the other hand he considered the capacities of Océ as a full-fledged replacement for a F/OSS community. The first point implies that the yanolc community cannot be assessed with the same measure as a large F/OSS project, the second point implies that the lack of community support doesn't matter because it is compensated by Océ's own resources.

6. AdaCore Couverture

6.1 Context

AdaCore, together with Open Wide, ENST and LIP6, is developing a Free Software coverage analysis toolset. In addition to the tools, the project aims to generate artifacts that allow the tools to be used for safety-critical software projects undergoing a DO-178B software audit process for all levels of criticality. The project follows the open source philosophy, and it is being moved to an open forge to allow for external contributions.

AdaCore wants to use the QualOSS methodology to analyze and understand what could help to make the Couverture project a robust and evolvable F/OSS endeavor while moving it to an open forge. Hints about how to create and interact with the new community, how to create documentation and procedures, and how to develop and organize the code would be very valuable.

6.2 Organization of the Assessment

The assessment method follows the assessment process prescribed by the QualOSS methodology. This means, that it conduct the following 5 tasks:

- Initiate the assessment
- Set up and plan the assessment
- Collect and analyze data
- Interpret the results of an assessment
- Supervise an assessment

The person in charge of doing most of the work of this assessment was José Ruiz who could have access to the internals of Couverture.

The people interviewed were Tristan Gingold (AdaCore) whose role is that of main developer of Couverture, and José Ruiz, whose role is that of user.

6.3 Couverture Assessment

The FlOSS use context of this assessment is that of an integration of a product (use of the Couverture toolset), the F/OSS collaboration context is Full F/OSS Collaboration (all partners and external contributors will collaborate in the development), the assessment mode is introspection (how to make the project Couverture evolve successfully in an open community), and the F/OSS endeavor scope is for a single F/OSS project (the whole Couverture project). The version of the assessment used is the Standard QualOSS Assessment, Version 1.0.

6.3.1 Scoping the Couverture Assessment

Couverture	
Description:	*The Couverture project: code coverage toolset for safety-critical software*
Official Website: • *http://www.projet-couverture.com* (Official project site) • *http://www.open-do.org/projects/couverture (Open Forge)*	
SCM Sites:	*svn://scm.forge.open-do.org/svnroot/couverture/trunk/couverture*
Issue Tracking System: • Internal to AdaCore • http://forge.open-do.org/tracker/?group_id=8 (public bugs, support, patches, feature requests)	
Mailing Lists:	• *Internal to AdaCore* • *http://lists.forge.open-do.org/cgi-bin/mailman/listinfo/couverture-discuss (Project Couverture general discussion: public)* • *http://lists.forge.open-do.org/cgi-bin/mailman/listinfo/couverture-patches (Couverture patches and contributions: public)*
Forums: • N/A	
Package Distribution Sites : • Only SVN access	
Prog Lang.	*Ada*

6.3.2 Work Products

Couverture is a Free Software coverage analysis toolset, being developed following the open source philosophy, and being moved to an open forge to allow for external contributions. It is very new and it is expected to be lacking things in terms of code, documentation, testing, and issue management. This section provides the results of the QualOSS assessment grouped by the characteristic examined by the QualOSS Standard Assessment.

6.3.2.1 Maintainability

Metrics related to issue/enhancement management were demanding in terms of resources (specific queries for the internal tracking system), showing that this is a very active endeavor where bugs are fixed quickly, and enhancements are assigned and implemented rapidly.

Metrics related to the code were measured using *GNATmetric*, showing a code with a slightly high complexity and pretty low coupling. Issues and enhancements are implemented very quickly.

The time spent to get the metrics and characteristics related to product maintainability was around 4 hours.

Characteristic	***Couverture***
Percentage of unassigned issues	0.00%
Average cyclomatic complexity per defined routine	3.94
Percentage of comments	2300 lines of comments 10213 lines of code 1131 total cyclomatic complexity
Evolution of number of lines of code between successive releases	N/A (no releases)
Average efferent coupling of low level modules	0
Average number of patch per issue	Evolving quickly
Evolution of change in code between minor releases	N/A (no releases)
Evolution of cyclomatic complexity of defined routines between successive releases	N/A (no releases)
Average efferent coupling of high level modules	0.34
Percentage of accepted enhancement proposals	100.00%
Evolution of change to public interfaces between minor releases	N/A (no releases)
Evolution of change in code between major releases	N/A (no releases)
Rapidity of implementation of enhancement proposals	2
Rapidity of issue resolution	7.59
Evolution of change to public interfaces between major releases	N/A (no releases)

6.3.2.2 Reliability

Metrics related to issue accounting, code and their evolution were not available for Couverture because there has been no release.

6.3.2.3 Security

There was not a single occurrence of vulnerabilities related to the Couverture toolset so there is no information that can be extracted (this is a very new project and not the type of software that is subject to this kind of issues).

6.3.2.4 Documentation

The documentation is not yet very mature, and there are many document types and a big amount of information missing.

The assessment of the documentation took around 5 hours.

Characteristic	*Couverture*
Document type availability	30.77%
Documentation information availability	10.08%

6.3.2.5 Tests

The testing procedures are not yet very well documented, and there is no much information to infer how it is applied. Just very light back-box testing is available so far.

The assessment of the testing part took around 4 hours, and the most time-consuming part was the code coverage analysis. The *gcov* tool was used for coverage.

Characteristic	*Couverture*
Likelyhood future test reports	No test reports (no release yet)
Test report availability	No test reports (no release yet)
Environment test availability	Only system tests
Unit test coverage adequacy	No unit test
System test coverage adequacy	44.00%
Unit test suite adequacy	No unit test
System test suite adequacy	No systematic testing and not well covered
Ease of testing	Not well documented

6.3.3 Community Members

There is a small community of users/developers, but it is increasing rapidly and the share of workload is appropriate.

The assessment of these software processes took around 6 hours.

Characteristic	*Couverture*
Evolution of new contributors (code)	Increasing
Evolution of new contributors (other than code)	Increasing
Evolution of new core contributors	Increasing
Core members stopping contributing	None
Core members number	Increasing
Longevity of commiters	8 months (new project)
Number of events	No events
Territoriality	60.14%
Lines per committer	623 lines per committer, 2 bugs per commiter
Percentage of handled files	9.10%

6.3.4 Software Processes

This is a very new endeavor, and software processes are neither well defined not implemented. The community knowledge is not formalized, and it relies on direct interactions among the developers (most of them work for AdaCore).

The assessment of these software processes took around 4 hours.

Characteristic	***Couverture***
Change submission maturity	Undocumented
Change review maturity	Undocumented
Change review adequacy	Undocumented
Committer promotion maturity	Undocumented
Commit review maturity	Undocumented
Commit review adequacy	Bug tracking and version control linked
Enhancement proposal maturity	Undocumented
Enhancement proposal adequacy	Undocumented
Issue management maturity	Undocumented
Issue management adequacy	Undocumented
Testing process maturity	Documented (for system testing) and followed
Testing process adequacy	Undocumented
Release planning maturity	Undocumented
Release planning adequacy	Undocumented
Release management maturity	Undocumented
Release management adequacy	Undocumented
Release backport maturity	Undocumented

6.4 Human Perception

People interviewed

- Tristan Gingold (AdaCore) whose role is that of main developer of Couverture
- José Ruiz (AdaCore) whose role is that of user

6.4.1 Perception of Couverture before the QualOSS assessment

Similar to the case of the GCC back-end, the two interviewees from AdaCore were again in a double role of AdaCore employees and F/OSS community members. The implications of this double role for the human perception of the quality of the F/OSS product and the results of the QualOSS Standard Assessment have been discussed above.

The pre-assessment of the quality of Couverture showed that, with regard to work products, good code quality and documentation were the most important requirements, together with the demand for working examples and tests. Regarding community-related quality aspects, the most desired features were a vivid community with expertise and an open communication style that allows discussions and attracts new people. Good community management and “external interest” (i.e. from businesses) were other important requirements in this respect. Regarding quality aspects related to the software process, clear guidelines and

policies on how to make changes (bug reporting, bug fixing) and a development plan for the different branches was required. Well defined automatic testing was a further requirement in this regard. The interviewees highlighted that at current the software process is largely determined by AdaCore and that external involvement in this process is only about to happen now.

Overall, the methodology of the QualOSS Standard Assessment was considered to be appropriate. The interviewees agreed on the structure of the measurement, the terminology, and its the scope.

6.4.2 Perception of Couverture after the QualOSS assessment

The perception of the measurement results of the two interviewees were quite similar. Tristan Gingold pointed out that Couverture is quite a young project. The fact that many results seem to indicate a high risk of Couverture for businesses is, according to Mr. Gingold, due to the fact that Couverture has no publicly available test suites in place yet.

José Ruiz emphasized that, to him, the results of the QualOSS Standard Assessment reflect very much the reality of the Couverture projects, especially that many things are missing at this early stage of the software. On the other hand, José Ruiz clarified, this circumstance results in the problem that a lot of information and data is missing. He saw a clear need for the Couverture community, as well as for AdaCore, to fix things, communicate better and to improve results. Especially with regard to businesses, he wants the overall indication of “quite a risk” to possible business users to change.

Tristan Gingold was positively surprised by the sheer number of measures that were checked by the QualOSS Standard Assessment. José Ruiz indicated that the assessment could be tailored to reflect internal (available to AdaCore but not to external people yet) information about, for example, more extensive testing and how to contribute to Couverture. Since both are available in-house at AdaCore it should be made public, so that a measurement platform like QualOSS can access these information.

Overall, both interviewees fully shared the quality assessment as provided by the QualOSS Standard Assessment.

7. Conclusions

The case studies described in this document cover a wide range of exercises to study the QualOSS methodology and its assessment method. From the various F/OSS endeavor acquisition scenarios described in deliverable D4.1, we cover two out of three F/OSS use contexts (integration in a product and in a service, but not in an infrastructure), two out of four collaboration contexts (the most important, Full F/OSS Collaboration, plus F/OSS Fork, but not Exploit or Takeover), the three assessment modes (product comparison, version comparison and introspection), and two out of three endeavor scopes (a complete project and a part of a project, but not a set of projects).

From the four case studies, three have been conducted on entities internal to the QualOSS consortium (AdaCore and Océ) and one on an external entity (Freecode). The advantage of starting with internal case studies is that we had much better access to the information and the people. It allowed us to refine both the assessment and the way interviews worked. After the required adjustments, the last case study was performed on an external endeavor (Asterisk) and on an external group (Freecode). This ensures the validity of the methodology, which could otherwise be biased by an internal-only validation.

The interviews that have been performed in order to evaluate whether or not the QualOSS Standard Assessment provides valid and reliable results indicate that the developed approach and methodology works very well for different acquisition scenarios. Overall, the results of the interviews can be summarized as follows:

- the structure, scope and level of depth of the measurement is appropriate; though some interviewees suggested to shorten the measurement they usually also saw that this would be aligned with severe limitations of the usability of the QualOSS Standard Assessment

- the large number of measures and indicators was often seen as an advantage because it allows to look at quality issues from many different angles, in a very flexible way
- the different measures and indicators were perceived to measure quality in a meaningful way
- suggestions were made to improve the measurement with a more advanced QualOSS Assessment methodology; the most important point in this context was to provide results together with more explanation
- the degree of user satisfaction with the QualOSS Standard Assessment on a scale from 1 (not at all satisfied) to 5 (very satisfied) ranges from 3.5 to 4.5.

The average time required for the assessments was under a week, giving pretty interesting indicators to people. Adapting the QualOSS assessment to the specific needs of the people asking for the evaluation (removing characteristics and indicators which are of no interest in a given context and adding specific tailored ones) would be an interesting option to increase the satisfaction and reduce the cost of the assessment.