Sponsored through Framework Programme Sixth (Call 5) by

Document Information

Version: 1.0
Date : Jan 19, 10
Pages : 15

Owning Partner:
ZEA

Author(s):
Xavier Heymans (ZEA)
Paul J. Adams (ZEA)

Reviewer(s):
Xavier Heymanns (ZEA)
Paul J. Adams (ZEA)

To:
European Commission

Purpose of distribution:
For E.C. review

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 - Indended for QualOSS consortium only
[ ] Confidential -individual partner only

Deliverable ID: D6.3

Title:

Exploitation Report

Deliverable: D6.3

Title: Exploitation Report

Executive Summary:

The purpose of this Exploitation Plan is to document the potential market for QualOSS results beyond the boundaries of the project itself. In particular, the manners in which QualOSS results can be maintained without EC funding are documented. Whilst not a formal business plan, this document helps to make the case for further exploitation of QualOSS results by providing information that would be commonly found in such a plan: basic market analysis, potential users and competing alternatives. In addition this content related to potential business opportunity around QualOSS results, this report also provides details of exploitation activities that have been conducted to date.

TABLE OF CONTENTS

1. Exploitation Opportunity 3
1.1 Users, Customers 3
1.1.1 Academic 3
1.1.2 Consultants 4
1.1.3 Free Software Community 4
1.2 Example Competing Solutions 4
1.2.1 OpenBRR 4
1.2.2 Ohloh (http://www.ohloh.net) 5
1.3 Description of QualOSS Quality Assessment Method 5
1.3.1 Unique Selling Points 5
1.3.2 Potential For Exploitation 8

2. Activities to Date 9
2.1 Case Studies 9
2.1.1 AdaCore/Gcc-backend Study 9
2.1.2 OSL/Yanolc Study 10
2.1.3 AdaCore/CoverageTool Study 11
2.1.4 Freecode/Asterisk Study 12

3. FOSS-ORI 13

1. Exploitation Opportunity

Enterprises increasingly acquire Free/Open Source Software (F/OSS) to integrate them in their products, whether software or hardware. As reported by Gartner in (Gartner, 2008), it is anticipated that 80% of all proprietary software application will include F/OSS code by 2012.

It is therefore important to provide a tool for enterprises to select the most appropriate F/OSS development endeavors to collaborate with. As enterprise mature in their use of F/OSS, they will understand the need to fully collaborate with the existing F/OSS community rather than be tempted to fork existing F/OSS code. In such a full collaboration mode, an enterprise will contribute bug corrections and eventual new features but it will benefit from others' contributions. When collaborating with a large F/OSS community, the benefit largely outweighs the effort provided by any single contributor.

To assess the worthiness to enter in a collaboration with a F/OSS endeavor, an enterprise will be interested to verify that the F/OSS code, the community, the software process, the libraries and tools show an acceptable quality and maturity. In other words, a F/OSS endeavor should be assessed for its robustness and evolvability where robustness means the capability to solve current issues and evolvability, the capability to continue to solve future issues (see Section 2 for a complete definition of robustness and evolvability).

QSOS (QSOS, 2006) and OpenBRR (OpenBRR, 2005), respectively sponsored by Atos Origin and Intel, both propose assessment method for assessing F/OSS project. However, they do not really assess the quality of the code. In general, these two methods tend to be too light to make a fully informed decision. The Std QAM has been built around the problem of F/OSS acquisition with the purpose of integrating it in a product hence the results likely have a much higher added value for this acquisition context than any other assessment methods.

<This text is lifted from Std QAM 1.1>

1.1 Users, Customers

1.1.1 Academic

As with all results from research-driven activities, one of the primary target audiences for exploitation of results is the academic community. Given enough interest in the topic, the academic community would continue to build upon the work of QualOSS. Building research ideas upon established work is a cornerstone activity of rigorous research and the open sharing of results is at the heart of this activity.

As funding can be a scarce resource and to make third-party evaluation of results easier, typically academic content is licensed in an open manner. Usually this is implicit, but it is becoming increasingly common for academics to formally release their results under a Creative Commons License. Implicitly or otherwise, academics typically license their content in a manner which allows free redistribution and derivation as long as the original author is identified.

So whilst the academic community is crucial to the on-going success of the work of QualOSS, it would be inappropriate to license the results in a manner which holds financial obligations for that use.

1.1.2 Consultants

Consultants act in the interests of their clients to help them make decisions regarding the software that they choose to use. Often, for certain types of software, consultants have a preferred choice for all situations. Sometimes there is no preferred choice and consultants will present multiple alternatives and advise on one.

In the latter of these scenarios consultants could use the QualOSS approach to quality assessment in order to provide structure to their decision making process and to provide extra information to their clients.

1.1.3 Free Software Community

The Free Software Community is a globally distributed network of businesses, individuals and other organisations which all contribute to create Free Software. Typically individuals are dedicated to a small number of Free Software projects (often volunteering their time), whilst businesses and other organisations might be involved within many projects.

The human resources within the Free Software community and within specific projects are often a scarce resource. This typically means that these people are responsible for too many activities within a project, or certain activities are simply not conducted.

By creating a structured approach to quality assessment, the Free Software community can benefit from the time saved by following QualOSS for the evaluation of their own software. This is as opposed to the heavyweight and/or ad hoc approaches that might already be followed.

1.2 Example Competing Solutions

Here, examples of competing solutions are briefly described: OpenBRR and Ohloh. Whilst this is by no-means an exhaustive comparison with the alternatives to QualOSS, this section provides and brief informational overview of the marketplace. The subsequent section will describe the unique aspects of QualOSS in greater detail.

1.2.1 OpenBRR

Overview: OpenBRR is a lightweight method for the quality assessment of Open Source software; this approach is largely focused on external quality characteristics. By weighting a selection of well-established metrics, the evaluator builds a picture of the “maturity” of the product and, thus, suitability for purpose.

Advantages:

- The effort required to make a complete assessment of a project is roughly ~2 man days.
- Low requirement on specialist tools; a complete assessment can be made using only a spreadsheet.

Disadvantages:

- Based upon external quality characteristics and so not easy to automate.
- Does not require traceability between data and results, thus discouraging audit of results.
- Not easily to visualize and, thus, compare results.
- No product (code) quality assessment.

1.2.2 Ohloh (http://www.ohloh.net)

Overview: Ohloh is an online directory of analytical data relating to Open Source development projects and the individuals who contribute to them. Visitors to the site can submit new projects for analysis, with no guarantee for the timeliness of obtaining results.

For any given project, Ohloh provides data according to simple measurements made against the source code: source lines of code over time, breakdown of the languages used in the project and breakdown of the licenses used. Ohloh also provides a

breakdown of activity per developer, detailing how many commits they have made and in what languages.

Advantages:

- The effort required to review the content on a particular project in Ohloh is minimal ~10 minutes.
- The processing of the source data to provide the facts and details is performed automatically. For larger projects, this could represent hours of processing requires significant processing power.

Disadvantages:

- Ohloh is not extensible; it is not possible for users of Ohloh to extend its functionality with new tools for data collection and processing.
- Ohloh does not provide assessment of the results. The target audience for Ohloh is the communities being studied themselves. Ohloh uses this to their advantage by leaving interpretation of the data and facts presented to the reader. This is less use, for example, to enterprises who need interpretation and not just numbers.
- Lack of traceability. Whilst Ohloh provides information on where and how they collect their data, they do not link their processed results and facts back to the exact source data that they have used.
- Ohloh have not released their processing tools s Open Source. This means that the results that they produce cannot be fully scrutinized and, therefore, trusted.
- Ohloh is currently a free service accessed through the web. This service is wholly-owned by a company who may decide to close down the service (either completely or to free access) at any time. It is, therefore, a risk to build Ohloh into any business or Open Source processes.

1.3 Description of QualOSS Quality Assessment Method

1.3.1 Unique Selling Points

The various efforts related to QualOSS may be categorized in two:

1. research specifically addressing FlOSS quality issues;
2. generic research on software product quality independent of whether the license is FlOSS or proprietary.

The former category is further developed below while research efforts in the latter category are only briefly mentioned 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 FlOSS quality issues, a distinction is made between (a) efforts that propose FlOSS assessment methods and (b) those that collect and process data with no intent to use the results for assessing FlOSS.

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 FlOSS. 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 FlOSS 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 FlOSS data collection is FLOSSMole. Its aim is to provide a database of information collected from well-known 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, FlOSS 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, FlOSS 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 FlOSS projects are missing.

In conclusion, due to the variable quality of FlOSS 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 FlOSS 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 FlOSS 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 FlOSS 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 FlOSS 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 FlOSS data. However, unlike FLOSSMole and FLOSSMETRICS, Ohloh has created a website to facilitate the viewing FlOSS 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 FlOSS 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.

1.3.2 Potential For Exploitation

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.

The following scenarios are presented in detail in the “Final Activity Report”.

- **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.

- **F/OSS Assessment Program**

This scenario is envisaged for companies who want to establish a F/OSS selection program for their whole enterprises.

- **Training for New Assessors**

Training on the various QualOSS results can also be an interesting way to become independent for an enterprise.

2. Activities to Date

2.1 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 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.

2.1.2OSL/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.

2.1.3 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.

2.1.4 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.

3. FOSS-ORI

Partners Involved: URJC, ZEA, UNU-Merit From: 05/09 To: 11/09

Description:

For the past decade open source has grown in popularity as a topic for researchers worldwide. The diversity of the open source ecosystem has presented research opportunities in software engineering, economics, management, legal issues and much more. Over this period hundreds of rese archers have produced results which directly benefit the open source communities being studied and yet these communities rarely gain access to these results in order to exploit them. Envisioned by Zea Partners, the Free/Open Source Software Open Res e arch Initiative was started in 2009 as a means to bridge this divide between the research community and open source communities. This will ensure the dissemination of research results internationally, to all aspects of open source communities: developers, users and SME service providers.

The FOSS-ORI does not aim to change the manner in which research projects disseminate their results. Instead, FOSS-ORI aims to supplement existing practices in order to account for certain features of modern research dissemination and provide a practical, effective solution to the fragmentation of dissemination effort between research projects.

Potential For Exploitation:

Whilst not, in itself, an exploitation activity, FOSS-ORI is an initiative which helps enable the exploitation of the research of the QualOSS consortium. The initiative involves contributions from QualOSS, FLOSSMetrics and SQO-OSS projects.

FOSS-ORI is specifically targeting the FLOSS contributor community and therefore there is little potential for financial gain from the initiative. The initiative is deliberately designed to be low maintenance and will be funded in the future through small contributions from future research projects associating with the initiative.