Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR and QSOS*

Jean-Christophe Deprez and Simon Alexandre

Centre d'Excellence en Technologies de l'Information et de la Communication (CETIC), Charleroi, Belgium
{Jean-Christophe.Deprez, Simon.Alexandre}@cetic.be

Abstract. Many organizations using Free/Open Source Software (F/OSS) are dealing with the major problem of selecting the most appropriate software product corresponding to their needs. Most of theses companies are currently selecting F/OSS projects using ad-hoc techniques. However, in the last couple of years, two methodologies for assessing F/OSS project have emerge, namely QSOS and OpenBRR. The objective of this work is, through a detailed and rigorous assessment methodology comparison, to allow companies to have a better understanding of these two assessment methodologies content and limitation. This work compares both methodologies on several aspects, among others, their overall approaches, their scoring procedures and their evaluation criteria.

Keywords: assessment methodologies, open source, free software.

1 Introduction

Many organizations have started to integrate Free (*libre*) Open-Source Software (F/OSS[1]) in their products, systems and infrastructures. Furthermore, software solutions that rely on F/OSS components become more frequently available to customers. In turn, organizations want assurance regarding the quality of F/OSS projects before integrating them in their solutions.

Having identified this need, several methodologies help select appropriate FlOSS projects have surfaced in the past couple of years. Two prominent methodologies are the Qualification and Selection Open Source (QSOS) backed by Atos Origin and Open Business Readiness Rating (OpenBRR) created by Carnegie Mellon West and Intel. Although fairly light weight, these two methodologies help shed lights on the seriousness of FlOSS projects.

The contribution of this paper is to compare the two assessment methodologies, OpenBRR and QSOS. In the end, it helps identify which of the two methodologies better fits one's context. Our comparison is performed based on the description of the methodologies and not on their empirical application. Based on our findings, our future work will aim at creating a new methodology, namely, the QUALOSS

* This work is partly funded by QUALOSS (#33547), a research project funded under the FP6 programme of the European Commission.

[1] F/OSS stands for Free *libre* Open Source Software.

A. Jedlitschka and O. Salo (Eds.): PROFES 2008, LNCS 5089, pp. 189–203, 2008.

methodology that takes advantages of the strong points of QSOS and OpenBRR while eliminating the weaknesses of both.

The rest of this paper is structured as follows. Section 2 presents the two methodologies. Section 3 introduces our comparison approach. Section 4 compares OpenBRR and QSOS. Section 5 reviews the advantages and weaknesses of both methodologies. Section 6 discusses the related work and Section 7 concludes with our future work.

2 Description of QSOS and OpenBRR

Both assessment methodologies help their users reach a similar goal, that is, select among a list of similar F/OSS projects for the one best suited to their context. The two following subsections present QSOS and OpenBRR respectively.

2.1 QSOS

This section presents version 1.6 of QSOS, which appears in [1]. The top part of the QSOS methodologies consists of the following series of steps:

- *Start from a list of F*lOSS projects whose products seems to fit with the overall requirements (set by the product user or product integrator)
- Evaluate each FlOSS project along the list of evaluation criteria given by QSOS. This step assigns an absolute score to each evaluation criterion.
- *The evaluator can adjust the importance of each criterion based on the particular context. This is done by assigning a weight and threshold to each criterion.*
- The scores obtained using 2 are weighted based on 3 so as to get the relative score corresponding to the given context. The outcome of this step is a ranking that determines the qualification or elimination FlOSS projects.
- QSOS then suggests trying the top FlOSS projects that meet qualification criteria. This number can vary between 2-5 FlOSS products depending on the criticality of the decision.

QSOS also provides a tree-hierarchy of evaluation criteria shown in Figure 1 along with a procedure to score each leaf criterion. QSOS splits its evaluation template in two sections: a generic section and a specific section. The generic section includes criteria that apply to all software products while the criteria of the specific section include an expected list the functionality and therefore varies according to software product family such as groupware, cms, database, etc. Due to space consideration, we only present evaluation criteria of the generic section. It is worth emphasizing that the templates listing functionality of software product families is specific to the web version of QSOS (at www.qsos.org) but it is not described in [1].

Figure 1 shows the tree hierarchy of evaluation criteria in QSOS's generic section. We note that this hierarchy comes from the paper version (.pdf) of QSOS 1.6 and not the web version of the template, which slightly differs. In particular, the web version lists the item “Independence of development” under the “Industrialized Solution” category whereas the paper version includes it “Intrinsic Durability”. Second, the web version elevated “Exploitability” as a top-level category where as the paper version includes it as a sub-category of “Industrialized Solution”. Finally, the paper version

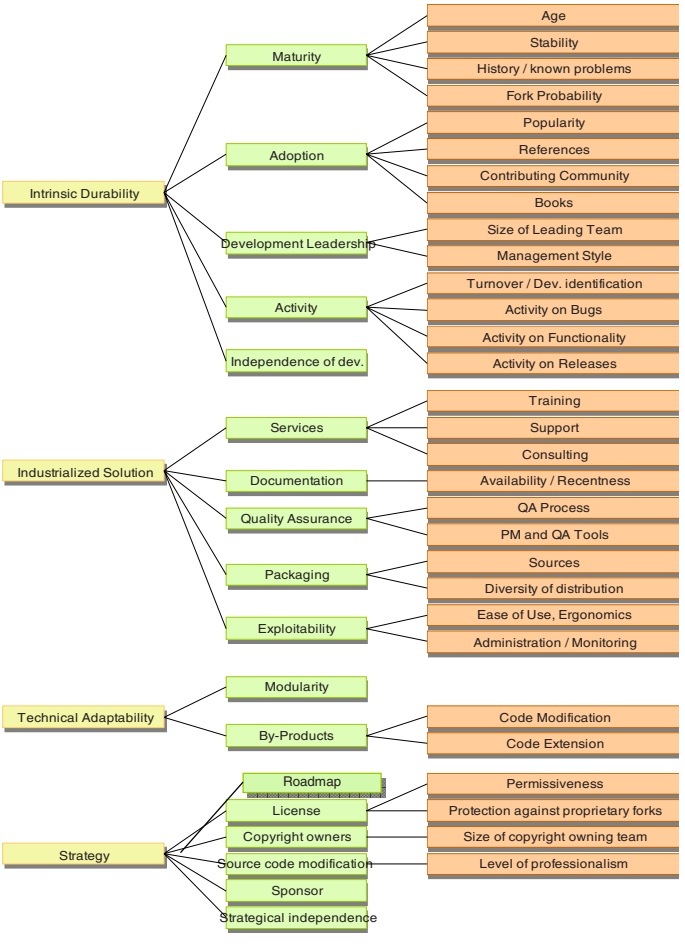

Fig. 1. Generic criteria from QSOS version 1.6

contains a section on "Service Providing" that list a few criteria to evaluate the ability of a service provider to help with the integration of a FlOSS component.

Along the hierarchy of Figure 1, QSOS gives a procedure to score each leaf criteria. A valid score is 0, 1 or 2. Table 1 shows a sample of the scoring procedure for three leaf criteria. We point to [1] for the description of the whole scoring procedure.

2.2 OpenBRR

OpenBRR [2] asks to follow these high level steps:

1. Perform a pre-screening (Quick Assessment). This step starts with a long list of F/OSS projects whose software product fits the overall requirement and ends with a few viable candidates for the final selection. This initial filtering is based on the criticality of the product or system that will integrate the F/OSS component as well as a handful of viability criteria determined based on the context.

Table 1. QSOS scoring procedure for three leaf criteria

Criteria	Score = 0	Score = 1	Score = 2
Age	Less than 3 month old	Between 3 month old and 3 year old	More than 3 year old
Training	No offer of training identified	Offer exists but is restricted geographically and to one language or is provided by a single contractor	Rich offer provided by several contractors, in several languages and split into modules of gradual levels
Source Code Quality	Not very readable code or of poor quality, incoherence in coding styles	Readable but not really commented in details	Readable and commented code implementing classic design patterns with a coherent and applied coding policy

2. Tailor the evaluation template (Target Usage Assessment): This step consists of reviewing and selecting the appropriate evaluation criteria from the hierarchy proposed by OpenBRR. One may also decide to add new criteria and procedure to score these criteria. The outcome is a customized version of the evaluation template fit to the context, which is usually defined by an organization, the kind of F*l*OSS software product to select, the criticality of the product in which the F*l*OSS component will be integrated, etc.
3. Data Collection and Processing: This step starts from the list of the F*l*OSS projects that passed the viability checks of point 1 above. It consists in collecting the raw data needed to score the evaluation criteria selected during point 2 and to apply the weight factor associated to each metrics determine during point 2 as well.
4. Data Translation: Aggregate the relative metric scores so as to obtain the score of each category. Each category score is then adjusted based on its weight factor determined during point 2. The final business readiness rating can then be published.

OpenBRR proposes a set of evaluation criteria for points 1 and 2 in our above list. In relation to point 1, the quick assessment, OpenBRR suggests looking at the following 8 issues and eventually adapting the list to better fit a given context:

(1) licensing, (2) standard compliance, (3) referenceable adopters, (4) availability of support, (5) implementation language(s), (6) third party reviews, (7) books, and (8) review by industry analysts such as Gartner.

For these criteria, OpenBRR let its user determine the scoring procedure and eventually what is a make or break situation. For example, if an organization only considers a F*l*OSS project when it is published under a license compatible with BSD then no use wasting time evaluating F*l*OSS components with a stronger copyleft bend.

Concerning point 2 above, the target usage assessment, OpenBRR proposes the template of evaluation criteria shown in Figure 2. In addition, OpenBRR gives a scoring procedure to evaluate the leaves of Figure 2. The procedure assigns a score between 1 and 5 where 1 is unacceptable and 5 is excellent. Due to space consideration, we only describe the procedure for three criteria in Table 2.

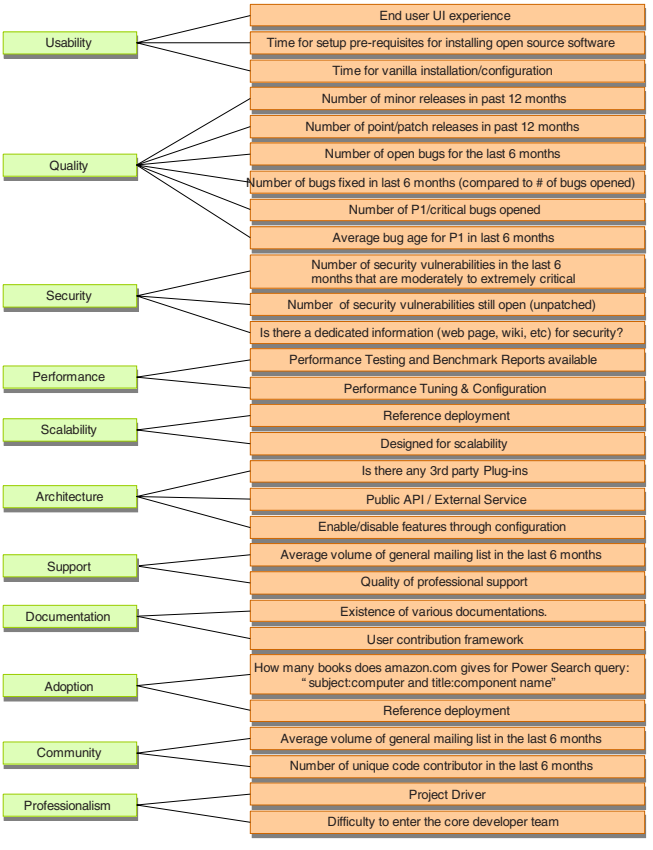

Fig. 2. OpenBRR hierarchy of evaluation criteria

Table 2. OpenBRR scoring procedure for three leaf criteria

Criteria	1	2	3	4	5
Time for vanilla installation	> 4 hours	1-4 hours	30min to 1 hours	10-30 minutes	< 10 minutes
User Contribution Framework	Users cannot contribute		Users are allowed to contribute		Users are allowed to contribute and contribution are edited / filtered by experts
Reference Deployment	No		Yes		Yes, with publication of user's size

3 Comparison Approach

Our comparison approaches is divided in several comparison exercises:

1. Comparison of the overall steps of the methodologies.
2. Analysis of the scoring procedures.

3. Coverage analysis of the evaluation criteria and adequacy of the hierarchies of criteria

The comparison of the overall steps first highlights the similarities and differences, including those related to how each methodology is intended to be applied in the field.

The analysis of the scoring procedures addresses the following three questions.

1. Is the range of scores adequate?
2. Is the scoring procedure of each evaluation criteria unambiguous?
3. Is the raw data required by the procedure in order to compute its result likely to be available in the field?

The coverage analysis compares the evaluation criteria between QSOS and OpenBRR in order to determine which are similar vs. those only present in only one of the methodologies. In addition, we also quickly assess the accuracy of the terminology used to express criteria and categories of criteria in each methodology.

4 Comparing QSOS, OpenBRR

In this section, we compare QSOS and OpenBRR based on the comparison approach introduced in Section 3.

4.1 Comparison of the Overall Approaches

This section compares the overall approaches by highlighting first, their similarities and second, their differences.

Both methodologies are similar in the following aspects:

- Each methodology proposes a predefined set of criteria for evaluating FlOSS projects. The set of criteria is categorized into a tree hierarchy of 2 levels for OpenBRR or of 3 levels for QSOS.
- The evaluation consists of scoring the various criteria based on a standard scoring procedure. During the evaluation of a given F*l*OSS project, this step results in assigning score to each criterion. We refer to this score as absolute.
- Users can adjust the importance of each criterion according to their context by varying the weight assigned to each criterion. In particular, during an evaluation, the absolute scores are weighted based on their importance to the current evaluation context. We refer to the weighted absolute scores as relative scores.
- A decision can be taken based on the resulting relative scores.

However, the 2 methodologies do differ in the order in which they apply the points above. The current order reflects that of QSOS while OpenBRR suggests inverting point 2 and 3 so that users first select criteria relevant to their context and therefore avoid scoring useless ones. In addition, OpenBRR allows the creation of new criteria as well as the tailoring of the scoring procedure for criteria.

These variations result from the difference in how each methodology is to be applied in the field.

QSOS believes that the absolute scores obtained when applying the scoring procedures are universal. Hence, the scoring procedure for a particular version of a F*l*OSS

project only takes place once. Others can then comment on the evaluation if a score seems unfair. However, once it is agreed on, the absolute scores of the given version of a F/OSS projects are universal and eventually made available to everyone. The only way to adjust the final outcome of an evaluation is to adapt the weights assigned to evaluation criteria.

OpenBRR on the other hand is a standard methodology but it assumes that every user instantiates it in a slight different way. Hence, the evaluation of a particular F/OSS project would result in slightly different scores depending on the context in which the evaluation is performed. The result of an evaluation is therefore not meant for reuse, even the absolute scores obtained for the evaluation criteria. In the case of OpenBRR, this seems a wise decision since the user is free to eliminate and add evaluation criteria as well as to modify and create new scoring procedures for new or existing criteria. In conclusion, OpenBRR provides a standard methodology that is expected to be tailored in practice.

As a result of the difference between the application and reuse strategies of QSOS and OpenBRR, it is easy to find a growing repository of F/OSS product evaluations for QSOS but not for OpenBRR.

The important question is: "Can a methodology be universal, at least partially such as QSOS, so that intermediate score (such as the absolute score of QSOS) can be re-used and shared?"

First, QSOS can apply such a strategy because its scoring procedure only allows a range of three scores 0, 1, and 2. Hence, the variance of scores between different evaluator is reduced. We delay further discussion on the range of scoring procedures for our next comparison exercise in the next subsection. Second, we believe that providing a one size-fits-all scoring procedure is not adequate for every criterion. For example, QSOS attributes a score based on the number of books published. Some F/OSS products address niche markets and the publication of books is very unlikely however, various technical articles in professional magazine may very well be of interest and a good substitution to the book published criterion. Although QSOS could allow such a tailoring in the scoring procedure, we have not seen it in practice, at least in evaluation sheet accessible via http://www.qsos.org/sheets/index.html.

Finally, it is worth noting that QSOS defines the scope of an evaluation based on the particular version of a F/OSS product while this information is not clear for OpenBRR. Hence, we may suppose that OpenBRR intend to leave that issue open so that it is possible for one to apply decide if the scope is a whole project or just a specific version of a particular F/OSS product. Leaving this decision to the users raises the following concern: When applying scoring procedures it is very important to clearly define the scope of the dataset so not to include data related to other versions. In some cases, this is not always easy or feasible. For example, the number of post on forums or the number of book published may not be about the particular version being evaluated. Hence, this may make the scoring procedure ambiguous.

4.2 Comparison of the Scoring Procedures

Both methodologies provide a scoring procedure in order to transform raw data into a score assigned to evaluation criteria. Table 1 and 2 respectively show a sample of the scoring procedures of QSOS and OpenBRR. Below we compare the complete scoring

procedures based on the 3 checks mention in section 3, in particular, the score range, the scoring procedure clarity/ambiguity, and the data availability.

Score Ranges

QSOS proposes a procedure whose result assigns a discreet score between 0 and 2, that is, 0, 1, or 2 to each evaluation criteria. Unfortunately, this range seems too restrictive to appreciate fully the information, at least, for certain criteria. We feel that a minimum of 4 levels would be required. With just 3 levels, the middle score may have true mid position but it may embed a positive or negative tilt.

OpenBRR has a procedure that assigns a discreet score between 1 and 5. In this context, a 5-level score is adequate. It is clear that 1 and 2 are negative while 4 and 5 are positive. Allowing a neutral score of 3 is also acceptable. However, we observe that in 14 of the 28 evaluation criteria, scoring rules do not use all 5 levels; in 13 cases, 3 of the 5 levels are used, in particular, 1, 3, and 5 skipping 2 and 4 and in the remaining case, 4 levels are used: 1, 3, 4, and 5. In turn, for more than half of the evaluation criteria, the procedure is no better than the 3 levels offered by QSOS.

Clarity and ambiguity of scoring procedures

Both scoring procedures lack clarity in certain of their scoring rules. We determined that the scoring rule of a criterion was ambiguous if the wording was unclear or if it could be interpreted differently by different people or in different contexts or also, if we identified a gap in the description of the scoring procedure, for example, the description given is clear for each score but there are many real-life situations not accounted for where it would be hard to settle on an actual score.

Our analysis finds the following:

QSOS contains a total of 41 evaluation criteria and 22 of them are found to be ambiguous. The 22 scoring rules we found ambiguous are for the following criteria: *stability, fork probability, popularity, references, contributing community, management style, activity on bugs, on functionality, on release, training, support, consulting, documentation availability, PM and QA tools, ergonomics, admin/monitoring, modularity, code modification, source code modification – level of professionalism, source code quality, intrinsic complexity, and technical documentation.*

The scoring procedure for *source code quality* is given in Table 1.

OpenBRR has 28 evaluation criteria. In most cases, evaluation criteria have much more specific meanings hence this leads to scoring rules that are much more precise. Nonetheless, we found 7 ambiguous cases: *end-user UI experience, is there a dedicated information for security?, performance testing and benchmarks, performance tuning and configuration, design for scalability, quality of professional support, and difficulty to enter the core development team.*

As already mentioned in Section 4.1, beside clarity, we can also question scoring rules on their range of applicability to the world of software products and components. However, we must be careful on the value of this comparison. OpenBRR recognizes that its rules may not be applicable to all situations hence allows tailoring by the evaluator. On the other hand, QSOS aims at providing scoring rules that compute universal scores hence it is more important for QSOS to propose generic rules. Incidentally, ambiguous rules usually seem more generic and it is thus the likely reason why more than half of QSOS's rules were found ambiguous. Furthermore, QSOS is capable to achieve a certain level of universality in its rules because its scores only vary between three discreet outcomes. Having a fourth or fifth outcome would make

it much harder for rules to stay generic. Unfortunately, as we pointed out earlier, a three point scale is likely not enough to truly help in good decision making.

Likelihood of data availability

In addition to the clarity of a rule, it is also important that the data requested be available otherwise the lack of data also put the applicability of the methodology at stake. When determining that some data is unavailable, it may mean that the data will be really hard to find but it can also suggest that the data is not readily available. That is, the raw data is not available and obtaining it would require posting a question on a forum hence would depend on the friendliness of community members and whether the members who answered actually know the correct data.

For QSOS, we found that 5 criteria have scoring rules requesting data unlikely to be available, in particular for the following criteria, *history/known problem, fork probability, management style, developer identification/turnover, independence of developments*.

For OpenBRR, we determined that 9 criteria asked for data that would likely be unavailable: *time to setup pre-requisites, time for vanilla installation/configuration, number of security vulnerabilities in last 6 months, number of security vulnerabilities still open, performance testing and benchmarks, performance tuning and configuration, reference deployment, designed for scalability, difficulty to enter the development team.*

4.3 Coverage of the Evaluation Criteria and Quality of Wording

This section first studies the similarities and differences among the evaluation criteria of QSOS and OpenBRR. This exercise is done for the leaf criteria of both methodologies. Second, we analyze the adequacy of the terminology used by both methodologies.

We start from the QSOS hierarchy and compare every leaf criteria with those of OpenBRR, including the viability criteria used in the quick assessment set, that is, the first pre-filtering step of OpenBRR.

The possible results of a pair wise comparison between two criteria A and B are:

- The two criteria are equivalent (A = B)
- One criterion is a more generic than the other (A < B (A is special case of B) or A > B (A is a more general case of B)),
- The two criteria have some similarity relationship of a fuzzy nature (A ~ B), for example, A may influence B or vice versa.
- The two characteristics have nothing in common

We must emphasize that our coverage analysis actually does not compare the criteria based on the semantic of their wording but rather compares them based on the semantic of their scoring rules.

In the comparison table shown in Table 3, the left column enumerate all QSOS characteristics and then, for every leaf characteristic of QSOS, we indicate whether OpenBRR has corresponding characteristic with one of the relationship signs identified above (=, < , >, or ~). This is shown by the relationship sign preceding the characteristics in the OpenBRR columns.

Table 3. Coverage analysis between QSOS and OpenBRR leaf criteria

QSOS			OpenBRR
Intrinsic Durability	Maturity	Age	NONE
		Stability	~ all Quality sub-criteria
		History, known problems (= Management Ability)	NONE
		Fork probability	NONE
	Adoption	Popularity	= Referenceable Adopters (from Quick Assessment)
		References (= level of mission criticality of references)	NONE
		Contributing community (= volume and diversity of community contribution)	> Community .. Average volume on general mailing list in the last 6 months > Community .. Number of unique code contributor in the last 6 months
		Books (number of books published about products)	= Adoption .. How many Books …
	Development leadership	Leading Team (= Size of leading team)	NONE
		Management style (= level of democracy of management)	~ Professionalism .. Project Driver ~ Professionalism .. Difficulty to enter core developer team
	Activity	Developers identification, turnover	~ Professionalism .. Difficulty to enter core developer team
		Activity on bugs	= Quality .. Number of open bugs, .. number of fixed bugs, and ..average bug age in the last 6 months + .. number of P1/critical bugs opened
		Activity on functionalities	NONE

Table 3. (*continued*)

QSOS			OpenBRR
		Activity on releases	= Quality .. number of minor releases and .. number of point/patch releases in past 12 months
	Independence of development		~ Professionalism .. Project Driver
Industrialized Solution	Services	Training (Diversity in geographical, cultural and gradual aspects)	NONE
		Support (Level of commitment assigned to support)	~ Support .. Quality of professional support
		Consulting (Diversity in geographical and cultural aspects)	NONE
	Documentation (Availability and recency of documentation)		~ Documentation .. Existence of various kinds of documentation
	QualityAssurance	Quality Assurance Process	~ Performance testing and benchmark reports available
		PM and QA Tools	NONE
	Packaging	Sources	NONE
		*nix packaging	NONE
	Exploitability	Ease of use, ergonomics	> Usability .. time for vanilla installation/configuration
		Administration/Monitoring (Availability of functionality for administration and monitoring)	NONE
Technical adaptability	Modularity (Software modularity)		~ Scalability .. Design for scalability ~ Architecture .. Are they any third party plug-ins? ~ Architecture .. Public API / External Service

Table 3. (*continued*)

QSOS			OpenBRR
	By-Products	Code modification (Ease of build-ability)	NONE
		Code extension (Extensibility or plug-ability)	= Architecture .. Are they any third party plug-ins? AND Architecture .. Public API / External Service
Strategy	License	Permissiveness	~ Licensing/Legal (in the quick assessment)
		Protection against proprietary forks	~ Licensing/Legal (in the quick assessment)
	Copyright owners (Size of copyright owning team)		NONE
	Modification of source code (Level of professionalism of procedure for proposition of modification.)		~ Professionalism .. Difficulty to enter core developer team
	Roadmap (availability + precision of the roadmap)		NONE
	Sponsor (Driving force behind product)		= Professionalism .. Project Driver
	Strategical independence		~ Professionalism .. Project Driver
Services Providing	Maintainability	Quality of Source Code (Volume of comment and use of design pattern)	~ Scalability .. design for scalability ~ Performance .. Tuning & Configuration (on user's end) ~ Architecture .. Are there any third party plug-ins? ~ Architecture .. public API / External service
		Technological dispersion (number of prog.lang. used)	~ Implementation language (in the quick assessment)
		Intrinsic complexity (Complexity of algorithms)	NONE
		Technical documentation (Design and arch doc + others)	~ Documentation .. Existence of various kinds of documentation

Table 3. (*continued*)

QSOS			OpenBRR
	Code Mastery	Direct availability (Number of experts available within a consulting company)	~ Support .. quality of professionalism support
		Indirect availability (Number of experts available in partner companies of serv. prov.)	~ Support .. quality of professionalism support

From Table 3, we observe that 16 QSOS criteria are not covered by OpenBRR. Conversely, we can also derive the OpenBRR criteria not covered by QSOS using Table 3 and Figure 2. In particular, the 7 following criteria are not covered by QSOS: end user UI experience, time for setup pre-requisites for installing open source software, all 3 criteria under security, reference deployment, user contribution framework. In addition, there are also 5 criteria from the quick assessment step of Open BRR not covered by QSOS: standard compliance, availability of a supporting or stable organization, implementation languages, third party reviews and industry analyst.

Beside our coverage analysis, we also add a few comments regarding the wording used by both methodologies for their criteria as well as for the higher-level nodes in their tree hierarchies.

We find that QSOS uses a very appropriate nomenclature for the higher level nodes in its tree hierarchies. However, the leaf criteria are usually summarized in very imprecise words. This forces the investigation of the scoring rules to understand accurately the meaning of criteria. For example, the criterion References under Intrinsic Durability .. Adoption is rather unclear. Once reading the scoring rules, we find that it measures the number of cases where users use a FlOSS product in mission critical solutions. Hence, the wording mission criticality of references would be more accurate without being too lengthy. This kind of re-wording could take place for several QSOS criteria.

For OpenBRR, this is the exact opposite. The wording of metrics is accurate and extensive although sometimes quite lengthy. Many criteria could therefore be re-worded in shorter phrases without loosing clarity. Concerning, the top node in the tree hierarchy, we find that the terms used are often very broad and inaccurate. For example, Quality is much too broad and a term such as stability or reliability would be more appropriate.

5 Advantages and Disadvantages of QSOS and OpenBRR

This section reviews the advantage and disadvantages of both methodologies. Besides helping decide which methodology to use, this comparison exercise will be our starting point to create a new methodology that preserves most advantages of both methodologies while getting rid of the disadvantages.

	Advantages	Disadvantages
QSOS	• Open repository of evaluation scores for various F/OSS projects (this pushes evaluators to collaborate on evaluation and to facilitate cross validation) • Extensive list of criteria • Interesting innovating nomenclature for the tree hierarchy • QSOS methodology is versioned and evaluation mention the QSOS version used	• Ambiguous scoring rules for more than half of the criteria • Scoring procedure with 3-level scale may make decision making harder • Universality of scoring rules is not possible for many criteria
OpenBRR	• Allows for tailoring hence better fit one's evaluation context • Clearer scoring procedure with fewer ambiguities • 5-level scoring scale for about half of the criteria • Ask evaluator to perform a quick assessment step to reduce the evaluation effort	• No open repository of evaluation (due to possible tailoring) • Does not exploit the 5-level scales for more than half of the criteria • Terminology is broad and imprecise for the top nodes in the hierachy • OpenBRR does not seem to be versioned. However, this may be left to the evaluator

In addition, we find that both methodologies have a particular important weakness. They do not require evaluators to capture the location of the raw data used to obtain the evaluation scores. This makes it hard to refute or argue the correctness of an evaluation. However, we did find that in practice, several QSOS evaluation sheets list URL's where raw data used for evaluation are mentioned.

6 Related Work

Prior to QSOS [1] and OpenBRR [2], other F/OSS evaluation methodologies were proposed, notably, two of them called Open Source Maturity Model respectively created by Golden from Navica [3] and by Frans-Willem Duijnhouwer from CapGemini [4]. In addition David Wheeler also proposed a very high level methodology to quickly evaluate F/OSS projects. These three efforts were used as a stepping stone by OpenBRR. On the other hand, OpenBRR and QSOS were created in parallel and to the best of our knowledge we are the first effort comparing F/OSS assessment methodologies.

An orthogonal body of research studies whether F/OSS development allows reaching software component of higher quality; an example of such efforts is found in [5]. These research endeavors have the objectives to determine how F/OSS development differs from the traditional methods used in the proprietary world and also to identify whether these differences impact the quality of code and of software products. On the

other hand, the evaluation methodologies compared in this work do not argue that F/OSS is better than proprietary. Rather, they give a mean to evaluate and to compare among several F/OSS alternatives without taking a stand on whether F/OSS or proprietary yields better quality.

On a more general note, the European Commission is currently funding several research projects related to open source and quality, namely, QUALOSS [6], FLOSSMETRICS [7], SQO-OSS [8], and QUALIPSO [9]. The first project listed is led by the authors of this article.

7 Future Work

Based on the comparison exercises presented in this paper, our future goal is to derive a new methodology for evaluating F/OSS projects. As OpenBRR learned from previous works, the QUALOSS project aims to bring F/OSS assessment to a higher level of objectivity, completeness, and clarity.

The goal is to create a methodology applicable at different level of thoroughness. A first light level will closely resemble QSOS and OpenBRR in principle with most of the advantages and without the shortcomings.

References

1. Method for Qualification and Selection of Open Source software (QSOS) version 1.6 © Atos Origin (April 2006), http://qsos.org/
2. Business Readiness Rating for Open Source © OpenBRR.org, BRR 2005 – Request fro Comment 1 (2005), http://www.openbrr.org
3. Golden, B.: Open Source Maturity Model © Navica, http://www.navicasoft.com/pages/osmmoverview.htm
4. Widdows, C., Duijnhouwer, F.-W.: Open Source Maturity Model © CapGemini (August 2003), http://www.SeriouslyOpen.org
5. Aberdour, M.: Achieving Quality in Open-Source Software. IEEE Software 24(1), 58–64 (2007)
6. QUALOSS (2008), http://www.qualoss.org/
7. FlossMETRICS (2008), http://flossmetrics.org/
8. SQO-OSS (2008), http://www.sqo-oss.eu/
9. QUALIPSO (2008), http://www.qualipso.org/