# Free/Libre Open Source Quality Models - a comparison between two approaches

Kirsten Haaland
UNU-MERIT
Maastricht, The Netherlands
haaland@merit.unu.edu

Arne-Kristian Groven
Department of applied research in information technology
Norsk Regnesentral
Oslo, Norway
Arne-Kristian.Groven@nr.no

Ruediger Glott
UNU-MERIT
Maastricht, The Netherlands
glott@merit.unu.edu

Anna Tannenberg
FreeCode AS
Oslo, Norway
anna.tannenberg@freecode.no

*Abstract*— This paper presents a comparison of a first generation software quality model (OpenBRR) and a second generation software quality model (QualOSS) by applying them to the case of Asterisk, a FLOSS implementation of a telephone private branch exchange (PBX, VoIP). Examining the trends and evolution of software quality models and identifying differences in the approaches and assessment outcomes, the results indicate significant progress in the development of open source quality models. However, it appears that tool support, which characterizes the second generation quality model, does not achieve to fully support the need for human interference. Therefore, future FLOSS quality models might call either for an even stronger reliance on tools and the abandonment of the human factor, or for an effective integration of both; the human factor and tools support. Effective, in this regard, means that the subjectivity aligned with human interference becomes marginal. At any rate, it appears that there is still a way to go.

***Keywords****: Free/Libre Open Source Software, FLOSS, quality models, Business Readiness Rating, OpenBRR, QualOSS, Asterisk 1.4.x, PBX, VoIP, quality metric, human perception.*

This paper has been made possible by EUX2010sec project, "Security infrastructure for the open source EUX2010 VoIP system", funded by The Research Council of Norway – project no. 180054/S10, and the EU funded project QualOSS, grant number 033547, IST-2005-2.5.5.

## I. INTRODUCTION

As software is increasingly becoming a core asset for any business, the correct selection and evaluation procedure for selecting business critical software is also gaining importance for every organization. Software quality models are geared to help in this decision-making. In the field of Free/Libre and Open Source Software (FLOSS), the fact that the source code is open and can be easily analysed, together with the fact that FLOSS project members communicate in a public manner, enables the assessment of software quality and the development of quality models.

This paper sets out with presenting the background and context of traditional software quality models, together with the evolution of major FLOSS quality models, notably differentiating first and second generation quality models. Section III outlines the measurement methodologies, and describes how they have been applied to Asterisk, a FLOSS implementation of a telephone private branch exchange (PBX, VoIP) (http://www.asterisk.org/). The actual results of the Asterisk measurements are presented and compared in section IV, identifying advantages and disadvantages of a first generation (OpenBRR) and a second generation (QualOSS) FLOSS quality model. Section V compares the assessment methods themselves, where congruences and discrepancies between a semi-automated quality assessment and the human perception of the quality of Asterisk are investigated. Finally, section VI concludes.

The questions addressed in this paper are:

1. What trends can be observed in the evolution of FLOSS quality models?
2. How does Asterisk perform in the two assessments and what can be learned from with regard to differences between the quality assessment of OpenBRR and QualOSS?
3. What conclusions can be drawn with regard to the quality and future of FLOSS quality models – do second generation models outperform first generation models?

## II. BACKGROUND AND CONTEXT

### *A. Traditional software quality models*

Quality is a very elusive concept that can be approached from a number of perspectives. Garvin [1] [2] have sorted out the various views of quality, the most important being: (i) *User view on quality*: Focusing on software that meets the users' needs. Reliability, performance/efficiency, maintainability and usability are core issues. (ii) *Manufacturing view on quality*: Focusing on conformance to specification and organizations capability to produce software according to the software process. Hence product quality is achieved through process quality. Defect count and staff effort rework costs are examples of relevant issues. (iii) *Product view on quality*: Focusing on specifying that the characteristics of products are defined by the characteristics (size, complexity, and test coverage) of its subparts. Component complexity measures, design and code measures all fall within this view.

Over the last four decades several approaches have been made to understand and control the quality of software products and their making. Roughly there are two main directions; "Quality management approaches" and "Quality model approaches". Within the first category of quality management, there is Crosby's quality management approach [3], Deming's quality management approach [4], Feigenbaum's approach [5] which is the TQM predecessor, and Weinberg's quality management approach [6]. Whereas the quality management approaches represent a more flexible and qualitative view on quality, the quality models represent a more fixed and quantitative [7] quality structure view. At least two directions of quality models exist. One direction is focusing around either processes or capability level, where quality is measured in terms of adherence to the process or capability level. Examples of such are all the variants of the proprietary Capability Maturity Model [8], CMM, including CMMI-SE/SW, ISO/IEC 15504 [10], or ISO9000 [11]. Another direction of quality models is focusing around a set of attributes/metrics used to distinctively assess quality by making quality a quantifiable concept. These include the McCall model [12], the Boehm model [13] [14], and the ISO9126 product quality standard [15], where ISO 1926 is based on Boehm's and McCalls

model's. To illustrate the structure of the latter type of model, the McCall model can serve as an example. McCall identified three main perspectives for characterizing the quality attributes of a software product.

These perspectives are:

- **Product revision** (ability to change).

  The product revision perspective identifies quality factors that influence the ability to change the software product, these factors are:

  - Maintainability - the ability to find and fix a defect.
  - Flexibility - the ability to make changes required as dictated by the business.
  - Testability - the ability to validate the software requirements.

- **Product transition** (adaptability to new environments).

  The product transition perspective identifies quality factors that influence the ability to adapt the software to new environments:

  - Portability - the ability to transfer the software from one environment to another.
  - Reusability - the ease of using existing software components in a different context.
  - Interoperability - the extent, or ease, to which software components work together.

- **Product operations** (basic operational characteristics).

  The product operations perspective identifies quality factors that influence the extent to which the software fulfils its specification:

  - Correctness - the functionality matches the specification.
  - Reliability - the extent to which the system fails.
  - Efficiency - system resource usage (including CPU, disk, memory, network).
  - Integrity - protection from unauthorized access.
  - Usability - ease of use.

Discussions around quality management approaches versus quality models indicate quality models advantages to be that they provide objective measurability and are simpler to use. Disadvantages of quality models are that they reduce the notion of quality to a few relatively simple and static attributes, and they represent leaner and narrower perspectives on quality.

### *B. First Generation FLOSS quality models*

Traditional quality models ignore various aspects of software unique to FLOSS, most notably the importance of community. This is not surprising since some models originate several decades back in time, when in the traditional software industry the focus was on firms, not considering the importance of the community or the interaction and dependence on outside expertise. Between 2003 and 2005 the first generation of quality assessment models emerged on the FLOSS scene, drawing on traditional models but being adapted to FLOSS. They were: (i) Open Source Maturity Model, OSMM Capgemini, provided under a non-free license. [16], (ii) Open Source Maturity Model, OSMM Navica, provided under the Academic Free License. (iii) Qualification and Selection of Open Source software, QSOS, provided by Atos Origin under the GNU Free Documentation License. (http://www.qsos.org/) and (iv) Open Business Readiness Rating, OpenBRR, provided by Carnegie Mellon West Center for Open Source Investigation, sponsored by O'Reilly CodeZoo, SpikeSource, and Intel, made available under a Creative Commons Attribution-NonCommercial-ShareAlike 2.5 License (http://www.openbrr.org/).

OpenBRR and OSMM Navica are in many respects similar, but they differ in the following: "The BRR model is more prescriptive, going into greater detail about what tests to carry out, and assigning specific scores to commensurable statistics while the OSMM leaves the scoring mechanism more open to interpretation" [18]. All models are based on a manual work, supported by evaluation forms. The most sophisticated tool support can be found for QSOS, where the evaluation is supported by either a stand-alone

program or a Firefox plug-in, which also enables feeding results back to the QSOS website for others to download. But still, the data gathering and evaluation itself is a manual work.

The status in 2010 is that none of these FLOSS quality models have seen a wide adoption, and none of them can really be considered a success, despite that the QSOS project shows a slow growth in popularity [18]. The OSMM Capgemini model has a weak public presence for the open source community [16], for the OSMM Navica model the web resource are no longer available, and the OpenBRR community consists of an abandoned web site that is frequently unavailable. The reasons for this lack of success is mixed, and probably a combination of the following: (i) The approaches have shortcomings, (ii) the knowledge about the approaches are not properly disseminated, (iv) the success stories are not properly disseminated, and (iv) the business expectations of the originators of these models were possibly unrealistic. However, even if there are shortcomings and lack of community support, the OpenBRR assessment model is still useful and influential enough to explore it further [19]. There are some success stories available, like the Open University's use of OpenBRR to select a Virtual Learning Environment [20], and also the fact that companies like FreeCode use it underlines OpenBRR's role. Further, the simplicity of a first generation model like OpenBRR is intuitively appealing and may have some advantages compared to second generation models.

### *C. Second Generation FLOSS quality models*

Recently, a wave of second generation FLOSS quality models has emerged. They all draw on previous methodologies, both traditional quality models as well as the first generation FLOSS quality models. The main difference is more extensive tool support. Second generation quality models are (i) the QualOSS quality model – a semi-automated methodology for the quality model drawing on existing tool support, explained in greater detail below (http://www.qualoss.org/), (ii) the QualiPSo OpenSource Maturity Model (OMM), a CMM-like model for FLOSS (www.qualipso.org/). QualiPSo OpenSource Maturity Model (OMM) "focuses on process quality and improvement, and only indirectly on the product quality". It is part of a large EC project initiative developing a Quality Platform for Open Source Software focusing on trust and quality in open source systems [21] [22]. The project aims to provide supporting tools and assessment process together with the OMM, which is still under development. It is a promising initiative and an interesting trend that a second generation quality model draws more strongly on traditional quality models, in this case CMM. The third second generation model is (iii) the SQO-OSS quality model – the Software Quality Observatory for Open Source Software (SQO-OSS) which is a platform with quality assessment plug-ins (http://www.sqo-oss.eu/). The quality model has developed the whole assessment platform from scratch, aiming at an integrated software quality assessment platform. It comprises a core tool with software quality assessment plug-ins and an assortment of UIs, including a web UI and an Eclipse plug-in [23]. The SQO-OSS is being maintained, but the quality model itself is not yet mature, and most focus is on the development of the infrastructure to enable the easy development of plug-ins.

## III. THE MEASUREMENT METHODOLOGIES APPLIED ON ASTERISK VOIP SOFTWARE

A comparison of a first and a second generation FLOSS quality model only makes sense if both models are applied to the same software. Asterisk is the world's most popular open source telephony project, under development since 1999. It is distributed under a dual license model, available under the free software license GNU General Public License (GPL) and a proprietary software license. The project and the community are directed by Digium, Inc., a US ICT company specialised in developing and manufacturing communications hardware and telephony software. Asterisk has more than two million users and is expected to be very mature in terms of code, documentation, testing, and issue management. The OpenBRR and QualOSS assessment of the quality of Asterisk form the basis for the empirical part of the paper.

### *A. The OpenBRR methodology*

A high-level view of the usage of the Open Business Readiness Rating (BRR) model to select the appropriate software to fulfill certain needs consists of the following three steps:

1. Creating a shortlist of software to be assessed.
2. Determining the relative importance of the categories and metrics.
3. Manually obtaining the data for the metrics.

Step 1 must of course be performed first, while step 2 and 3 may be performed in any order. It is a manual process, and it aims to be complete, simple, adaptable and consistent. A spreadsheet template is used when creating a BRR. The template used for the Asterisk BRR evaluation is an updated version provided to the authors by Dr. Wasserman from the Center for Open Source Investigation at Carnegie Mellon West, who founded the OpenBRR model in 2005, together with a number of high profile open source actors like Intel Corporation, Spike Source and O'Reilly Code Zoo. The main change is that the new template has only 7 categories (see Table I), compared to 12 in the old version. The OpenBRR method uses weights to provide flexibility. Weights are set both on categories and on individual metrics to ensure that a BRR can be easily adapted to the needs of different businesses [24].

First it was checked whether there were any available BRR's on Asterisk or other software of the same component type. However, there was no BRR for Asterisk or any other PBX available for referencing, and the following workflow was implemented:

**Step 1 - Filling out basic information** (sheet 1 and 2)

Name and version of the software to be rated:

- Asterisk v1.4.25

Names of the resources contributing to the BRR (Component type):

- PBX (VoIP)

Usage setting:

- Mission critical use (The possible Usage Settings are Mission critical use, Routine use, Internal development and/or Independent Software Vendor and Experimentation. As a PBX in use is likely business critical, Mission critical use was chosen).

**Step 2 – Setting category weights**

The OpenBRR analysis was applied to Asterisk version 1.4.25. The usage setting chosen for this evaluation was “mission critical use”. The category weights should be based on a business case, which unfortunately was lacking when the Asterisk rating was done. Setting category weights becomes a random task without a business case, but the assumption was made that Service and Support is very important for any business considering implementation of Asterisk or any other PBX, and so that category was given a high weight. Further, the weights were fairly evenly distributed between the categories, except for “Functionality” which was given a high weight as that is standard practice in most BRR's. The categories and actual weights are listed in Table I.

TABLE I. OPENBRR CATEGORY WEIGHTS FOR ASTERISK BRR

| Category name | Description | Weight |
|---|---|---|
| Functionality | Whether the software offers certain features | 25% |
| Operational Software Characteristics | Metrics concerning user experience, security, performance and scalability | 15% |
| Service and Support | Metrics describing availability of professional and community support | 25% |
| Software Technology Attributes | Metrics describing technical architecture, release cycle and code quality (bug statistics) | 10% |
| Documentation | Metrics describing the availability and quality of documentation | 10% |
| Adoption and Community | Metrics describing the activity of the community and existence of reference installations | 10% |
| Development Process | Metrics for stability and quality of project driver and code contributors | 5% |

Note that the combined weight of all metrics within each category is 100%. As there currently are no BRR's for other PBX's available, the weights were evenly spread among the metrics.

**Step 3 - Obtaining metric data**

There are 27 unique metrics to provide data for, excluding the Functionality category. Two metrics are used in more than one category. The OpenBRR model does not provide any tools for data mining so all data must be collected manually. The data was transferred into the BRR, resulting in a score for each metric from 1 to 5, where 1 is “Unacceptable” and 5 is “Excellent”.

***1. Operational Software Characteristics***

This category collects metrics concerning user experience, security, performance and scalability. All metrics pertaining to user experience are highly subjective and the score depends on both the stack on which Asterisk is installed and the experience of the person doing the installation. In this case, it was assumed that there were no platform limitations, and that Asterisk could be installed in the simplest way possible, through a Linux package handler or by installing AsteriskNOW, AsteriskNOW is the customized Operating System that comes with Asterisk bundled. The user was assumed to be a fairly competent system administrator without prior experience of Asterisk. Testimonials from Asterisk users found on the official website were also used in an attempt to make the Usability scores more legitimate. Information for answering metrics on Performance and Scalability was found by combining information on the official website www.asterisk.org and the wiki on www.voip-info.org with testimonials from users. Googling for Asterisk+peformance+benchmark also turned up some semi-interesting results. The source for information on number of critical security issues, www.secunia.com, is not an absolute authority but has been used for that metric in BRRs of other software products. Asterisk developers also issue Security advisories on the official Asterisk web page which were included in the raw score.

***2. Service and Support***

The Service and Support category contains only two metrics; the activity on the mailing list(s) and the quality of professional services for the software. A brief look at the official Asterisk web page told the rater that Digium Inc. provides a variety of professional support and services and that there are also service partners around the globe for customers looking for support close to their own location. Estimating the mailing list activity required some manual work. Asterisk mailing lists use Mailman. The main mailing list is called asterisk-users but there are several other mailing lists and also an active forum. There was no easy way to obtain mailing list statistics, so the number of messages were manually counted for the 6 most recent months. As the result was well over the threshold for the highest score, - including messages from other lists or the forum was not considered.

***3. Software Technology Attributes***

Information on Asterisk Architecture, including APIs and add-ons, was easily obtained from the official resources, webpage and wiki. Google was also used to find extra information on the different API’s that exist. It should be noted that it is not always completely clear which add-ons work with which versions of Asterisk. The BRR authors did not install all add-ons to make sure that they worked with the Asterisk version being rated. Data for the release cycle metrics in Quality was found by looking through the file ChangeLog in the Asterisk 1.4.25 package and counting the number of releases in the past year. 1.4.x-releases were considered minor releases, while 1.4.x.y-releases were counted as patch releases. The Asterisk project's bug tracker, Mantis, was used to answer the metrics related to bugs. Exactly how some of the bug metrics should be calculated and the definition of a P1/critical bug is not entirely clear from the OpenBRR model description and Mantis is not suited for running all the necessary queries. The scores for the bug metrics should therefore be seen as somewhat uncertain. These metrics have often been left blank in the sample BRR's that are available on www.openbrr.org.

***4. Documentation***

The Documentation metrics were easily answered as there are extensive documentation resources available on, or from, the official website.

***5. Adoption and Community***

In the Adoption and Community category, the result from the mailing list activity metric is reused. The number of Asterisk books on Amazon.com were found through an advanced search, not the suggested power search, which is outdated and returns no results. Digium presents a large number of reference deployments on

the official web pages and the metric asking for references was easily answered. The number of unique code contributors for Asterisk version 1.4.25 were found through a summary page in the package, listing all code contributors, some lacking commit rights and therefore not visible when looking at the list of subversion committers.

***6. Development Process***

The category Development Process contains two metrics; Project Driver and the recruitment of new core developers. The project driver is clearly Digium Inc, a single but strong corporation like MySQL. The recruitment of new core developers is not explicitly described, but in order to get commit access to Subversion, one must send in a number of patches of good quality, not anyone may commit code. Most of the core developers are Digium employed, but that is no requirement. This information was found on the webpage under the menu option Developers. All in all, it seems Digium and Asterisk have a good balance between inclusiveness and the level of quality they expect from their developers.

**Step 4 – Set metric weights**

All metrics have weights associated with them. The total metric weight within each category is 100%. If the BRR is to be compared to other BRR's of software components of the same type, the weights must be exactly the same for meaningful comparison. As there were no BRR's for other PBX's available and no business case or customer with specific whishes available when doing the rating, the weight was evenly spread among the metrics.

**Step 5 – Choose a feature set and get data for Functionality**

The Functionality category collects a set of features that the software product may have. What features are included is up to the rater to decide, but when comparing two products from the same software category, one must of course choose the same set. The features may be Standard Functionality, in which case a penalty is given if the product does not contain one of the listed features, or Extended Functionality, in which case the product will be awarded extra points if the functionality exists, but no points will be deducted in case the functionality does not exist. As per usual with the OpenBRR model, weights are set on the features.

When choosing feature set for the Functionality category, you would normally have a business case and actual requirements as basis. For the Asterisk BRR, the feature set and weights were chosen by two experienced Asterisk consultants and based on their opinions of what a PBX should provide in terms of supported codecs, protocols, hardware and functionality like GUI and conferencing. The aim was to collect as general a feature set as possible. However, choosing a feature set will always be a subjective endeavour and the details of the feature set should always be reviewed when BRR's are used to select a software product.

**Step 6 – Quality Assurance**

Quality assurance should always be performed on a BRR before it is considered completed. Errors in the formulas of the spreadsheet can easily be introduced, weights miscalculated, information sources excluded, etc. Quality assurance of the actual content is harder to perform as it basically means redoing all the work of the original rater. The content of the Asterisk BRR was quality assured by an experienced Asterisk user and his comments noted, if not worked into the BRR.

After setting the weights, the metric data was manually obtained for the 27 unique metrics. For “Functionally” a feature set had to be chosen. When choosing feature set for the Functionality category, one would normally have a business case and actual requirements as basis. For the Asterisk BRR, the feature set and weights were chosen by two experienced Asterisk consultants based on their opinions of what a PBX should provide in terms of supported codecs, protocols, hardware and functionality like GUI and conferencing. The aim was to collect as general a feature set as possible. However, choosing a feature set is a subjective effort and the details of the feature set should always be reviewed when BRR's are used to select a software product.

*B. The QualOSS methodology*

Like OpenBRR, QualOSS provides a high level methodology to benchmark the quality of open source software. Key criteria for the benchmarking are the evolvability and robustness of FLOSS [25]. Product characteristics, community characteristics and software process characteristics are considered to be equally important for the quality of a FLOSS endeavour.[1] The QualOSS assessment can be applied to both complete FLOSS products and components.

Overall, QualOSS assesses the quality of a FLOSS endeavour from a business point of view. QualOSS is based on the assumption that FLOSS quality is highly depending on the context in which it is used and the purposes a company pursues with it. QualOSS therefore distinguishes between usage scenarios and business scenarios. Regarding usage scenarios, several FLOSS endeavors can be compared for the purpose of selecting the most suited one (comparison scenario) or an actor in a FLOSS endeavor wants to monitor the robustness and evolvability of his FLOSS endeavor (introspection scenario) [26]. Business scenarios are determined by the way a company that wants to examine a FLOSS endeavour and how they deal with this endeavour. The business could fully cooperate with the FLOSS endeavour (e.g. communicate with the community and submit patches), it can exploit the FLOSS endeavour (e.g. using FLOSS components but giving no feedback to community), it can fork the FLOSS endeavour (i.e. creating its own independent version of the software), it can take over the FLOSS endeavour (i.e. control and steer the development), it can implement FLOSS components on its infrastructure, or it may want to sell a service on FLOSS [26].The scope, purpose and meaning of the results of a QualOSS assessment vary depending on these scenarios. For instance, a company that would like to fork a FLOSS endeavour pays less attention to the community and software processes than a company that aims at a full FLOSS collaboration.

The strong focus of the QualOSS assessment on context dependence of quality assessments also becomes manifest in considering quality issues from three different personal perspectives: the product manager perspective, the project manager perspective, and the architect, analyst and developer perspective.

The (maximum) scope of the assessment is illustrated in Figure 1, but an assessment can cover also cover a subset of the quality characteristics presented in the figure.

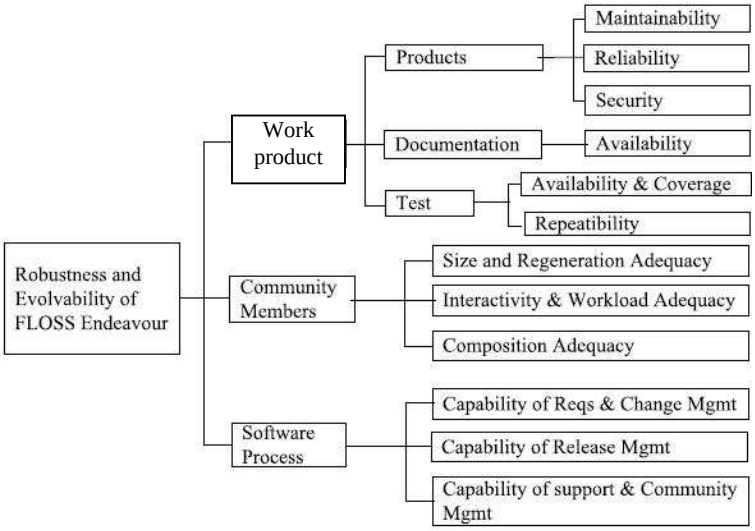


*Figure 1: Structure of the QualOSS Standard Assessment*

QualOSS uses the GQM, Goal Question Metrics, invented by Basili in 1992 [28] associating GQM templates with each of the leaf characteristics in the figure above, which are “Maintainability”, "Security", "Reliability", "Availability", "Availability and Coverage", "Repeatability", "Size and Regeneration Adequacy", "Interactivity and Workload Adequacy", “Composition Adequacy”, "Capability of Requirements and change management", “Capability of Release Management”, and "Capability of Support and Community Management".

[1] “FLOSS Endeavor is defined by 1) a set of work products, 2) the FLOSS 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”[25].

When applying the GQM, goals are formulated by specifying the issue addressed, the context (of validity), the view point, the quality focus, the object of the assessment, and the purpose of the assessment. The purpose of all assessment goals is to evaluate the degree of risk for a leaf characteristic, for particular viewpoint and for the selected context. GQM decompose the overall goal into assessment sub-goals.

The next step of GQM is to identify a series of questions for each assessment (sub-)goal. The combined answer to these questions determines the degree to which an assessment goal is satisfied. For an assessment goal on maintainability from a product manager's viewpoint, the questions are:

- How is the percentages of enhancements proposal that get accepted?
- How is the rapidity with which accepted enhancements are implemented?
- How is the percentage of changes in the code between major releases?
- How is the percentage of changes to public interfaces in the code (external API) between major releases?
- How is the evolution in code volumetry between various releases of the code over time (in chronological order)?

In QualOSS for all viewpoints and for each leaf characteristics of the quality model in Figure 1, a set of questions are predefined. The next step of GQM is to determine how to answer each question and how answers can be combined. The GQM also suggests that answer to questions should be done using sound data analysis and sound measurements. Actually, measures are combined into risk indicators. In the QualOSS approach, indicators are being developed to quantify the perceived risks associated to an assessment goal. For instance, a predictable behavior in a FLOSS endeavor will be perceived as less risky than unpredictable actions, even if such unpredictable actions may sometimes generate great outcomes.

The QualOSS assessment is a highly automated measurement and uses a number of measurement tools, such as CVSAnaly (http://cvsanaly.tigris.org/) and Bitcho (http://tools.libresoft.es/bicho ). However, a good proportion of the measurement, especially on documentation, has to be done manually. The measurement results are documented in a spreadsheet that is filled automatically and manually.

Like with OpenBRR, the basis for defining the underlying thresholds is experience-based rather than sound statistical analyses. This may be arbitrary to some degree, but it also allows for adjusting these thresholds to individual requirements. As described above, the QualOSS assessment focuses on three main dimensions to assess the robustness and evolvability of software: work products, community, and software processes. The following sections illustrate how Asterisk 1.4.26 has performed in these dimensions, providing the results of the QualOSS assessment grouped by characteristic [26].

- **Work Products**

QualOSS differentiates three categories of work products-related quality aspects: product, documentation and test. Quality characteristics at the product level are evaluated with regard to maintainability, reliability and security. Each of these criteria is made up of a number of detailed indicators and metrics. “Products” constitutes maintainability, reliability and security:

- *Maintainability* is a composite of 15 indicators, such as percentage of unassigned issues, percentage of comments, evolution of the number of lines of code between successive releases, average number of patch per issue, measures of cyclometric complexity and efferent coupling, and the like. The assessment results indicate that the Asterisk code is complex, changes extensively, and features a relatively low amount of comments. However, coupling is low, and issues are fixed very quickly. Metrics related to issue/enhancement management could not be extracted reliably from the bug tracker. The system used Mantis, which does not support the extraction of the required information in an automatic way. Therefore, issue/enhancement information was extracted manually, which took a reasonable amount of time and effort because there were not many instances to look at manually. Overall, the QualOSS assessment concludes, from a business point of view, that Asterisk 1.4.26 constitutes a medium risk with regard to maintainability.
- *Reliability* is composed of measures for stability evolution, the importance of corrections, and violations of code conventions. It turned out that the amount of issues in Asterisk is very high. Overall,

the risk assessment seems to indicate a small risk from a business point of view, but it must be taken into account that the QualOSS assessment of reliability remained fragmentary due to lack of information and limited applicability of tools. Metrics related to issue accounting and its evolution were problematic, because the bug tracker 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. Another restriction on the application of the QualOSS tools were provided by language issues. There is no tool that allows measuring violations of code conventions when the code is written in C, which is the case for Asterisk, so these metrics and indicators are not available.

- *Security* is composed of 9 indicators, all related to entries in the National Vulnerabilities Database. Asterisk is a kind of software typically 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; however the number of severe ones is relatively high and increasing over time. Overall, from a business point of view QualOSS constitutes a small risk from the point of view of security.

Altogether, these three aspects form the *product related quality and risk aspects* of Asterisk 1.4.26 for businesses. In total, Asterisk as a product constitutes only a small risk for businesses.

- *Documentation* consists of six indicators, related to the types of available documentation (e.g. manuals) and the information that is provided by these documents. Documentation indicators are the only QualOSS assessment indicators that use weights. The analysis turned out that many 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 large amount of knowledge embedded in mailing lists and in the developers themselves. Therefore, the overall result of the assessment of documentation is that it constitutes a medium risk for businesses.
- *Tests* is composed of 8 indicators, such as test report availability, unit and systems test coverage adequacy, ease of testing or likelihood of future test reports. The QualOSS assessment showed that testing procedures are not formally defined and that Asterisk relies extensively on human test effort, with few automated testing. Apparently, this FLOSS endeavour expects its community to act as testers. Overall, with regard to tests, Asterisk 1.4.26 constitutes a high risk for business.

Overall, the QualOSS assessment shows that Asterisk 1.4.26 constitutes a medium risk for businesses at the level of the *work product*.

- **Community Members**

Quality characteristics at the level of community members are composed of three indicators, size & regeneration adequacy, interactivity & workload adequacy, and composition adequacy. For technical reasons, the QualOSS assessment could cover only the first two of these indicators. The composite evaluation of these two indicators, which consists of factors such as the evolution of community members that report bugs or contribute code or other things to the endeavour, the evolution of core contributors, the longevity of committers, the evolution of number of commits, or lines per committer. Tha QualOSS assessment of these indicators has revealed that the Asterisk community is very large and more or less stable, with people doing very local changes. Asterisk is a mature endeavor, and software processes are rather well established, except for the testing (which relies strongly on manual testing). Overall, *size & regeneration adequacy* constitutes a negligible risk for business, whereas *interactivity & workload adequacy* constitutes a medium risk. Consequently, the *community members*-related quality of Asterisk 1.4.26 constitutes a small risk.

- **Software Processes**

Quality characteristics related to software processes are the *capability of requirements and change management* and the *capability of release management*, the *capability of support and community management* had to be excluded from the assessment for technical reasons. The composite result of the two testable characteristics, which are largely composed of indicators related to review maturity and review adequacy but also aspects of committer promotion, indicates that Asterisk 1.4.26 constitutes a small risk

regarding the *capability of requirements and change management* but a medium risk regarding the *capability of release management*. Altogether, these results denote that, at the level of software processes, the quality of Asterisk 1.4.26 constitutes a small risk for businesses.

## IV. Presenting and Comparing the Measurement Results

### *A. OpenBRR measurement results*

An overview of the OpenBRR assessment results is shown in the table below. The score for each metrics range from 1 to 5, where 1 is "Unacceptable" and 5 is "Excellent".

TABLE II. OpenBRR Results

| (sub-)category | Metric | Score |
|---|---|---|
| Functionality | N/A | 3 |
| Operational Software Characteristics | | |
| Usability | End user UI experience | 3 |
| | Time for setup pre-requisite for installing open source software | 5 |
| | Time for vanilla installation/configuration | 4 |
| Security | Number of security vulnerabilities in the last 6 months that are moderatly to extremely critical | 4 |
| | Number of security vulnerabilities still open (unpatched) | 5 |
| | Is there dedicated information (web page, wiki, etc) for security? | 5 |
| Performance | Performance Testing and Benchmark Reports available | 3 |
| | Performance Tuning & Configuration | 5 |
| Scalability | Reference deployment | 5 |
| | Designed for scalability | 5 |
| Service and support | | |
| | Average volume of general mailing list in the last 6 months | 5 |
| | Quality of professional support | 5 |
| Software Technology Attributes | | |
| Architecture | Are there any 3rd party Plug-ins | 5 |
| | Public API / External Service | 5 |
| | Enable/disable features through configurations | 5 |
| Quality | # of minor releases in past 12 months | 1 |
| | # of point/patch releases in past 12 months | 3 |
| | # of open bugs for the last 6 months | 4 |
| | # of bugs fixed in last 6 months (compared to # of bugs opened) | 5 |
| | # of P1/critical bugs opened | 2 |
| | Average bug age for P1 in last 6 months | 1 |
| Documentation | | |
| | Existence of various documents | 5 |
| | User contribution framework | 5 |
| Adoption and Community | | |
| Adoption | How many books does amazon.com give for Power Search query: "subject:computer and title: component name" | 5 |
| | Reference deployment | 5 |
| Community | Average volume of general mailing list in the last 6 months | 5 |
| | Number of unique code contributors in the last 6 months | 4 |
| Development process | | |
| | Project Driver | 4 |
| | Difficulty to enter the core developer team | 5 |

The rating for Asterisk 1.4.25 was completed on May 24th 2009 and the final score was 4.24. Including quality assurance, approximately 12 hours were spent on completing this BRR.

*B. QualOSS measurement results*

As illustrated in Table III, the composite result of the QualOSS quality and risk assessment denotes Asterisk 1.4.26 as a medium risk for business.

TABLE III. RISK ASSESSMENT TREE FOR ASTERISK 1.4.26

| | | | |
|---|---|---|---|
| Robustness and Evolvability of the Endeavour AVG 1.873 | Work product AVG 1.32 | Product AVG 2.216 | Maintainability AVG 1.596 |
| | | | Reliability AVG 2.1 |
| | | | Security AVG 2.682 |
| | | Documentation AVG 1.333 | Availability AVG 1.333 |
| | | Test AVG 0.5 | Test Availability and Coverage AVG 0.5 |
| | | | Test Repeatability AVG 0.5 |
| | Community members AVG 2.282 | | Size and Regeneration Adequacy AVG 3 |
| | | | Interactivity and Workload Adequacy AVG 1.563 |
| | Software processes AVG 2.017 | | Capability of requirements and change management AVG 2.333 |
| | | | Capability of release management AVG 1.7 |

Legend

| High risk | Medium risk | Small risk | Negligible risk |
|---|---|---|---|
| [0,1[ | [1,2[ | [2,3[ | [3,4] |

*C. Discussion of Results*

Discussing the results one should look for consistency in findings and results, whenever the two approaches provide measurements of comparable entities. Differences in coverage should be discussed and one should try to explain inconsistencies if they are present. Finally, one should also try to go beyond the measured results whenever necessary and if additional information is available.

Qualoss and OpenBRR both cover different views of quality, (i) the product view on quality, (ii) the manufacturing, or process, view on quality, and also to some smaller extent (iii) the user view on quality. But the differences in the two approaches are obvious: While OpenBRR is performed manually, having only a spreadsheet for registration of results and calculation of scores, the QualOSS model relies on more automation using software tool support to capture data on the Internet. But QualOSS also have to rely on manual labour whenever proper tools are not available, as it was the case for some of the measurements in the Asterisk example. There is also a difference in the output of the two quality assessment models. While OpenBRR outputs a score, QualOSS also outputs trend indications.

A general weakness in both assessments is the lack of fine-tuned real-world business contexts when performing the assessments. This is not say that the assessments were unrealistic, and some of the evaluators had deep knowledge in the technology to be assessed. But some assumptions had to be made by the evaluators, and the level of contextual detail could have been more fine-grained. In the case of OpenBRR, it is assumed by the model that there is a real need and specific business case as basis for the rating to answer questions like: Who is the customer? What are his/her needs? What is the level of technical knowledge of the customer? What is the available or preferred technical platform to run the software on? Without the answers to these questions, the final score becomes too much a product of the opinions and assumptions of the raters, especially obvious when choosing functionality set, evaluating the user experience, and of course setting all the weights for relative importance. The QualOSS assessment did choose to tune the evaluation towards the needs of a company making business of Asterisk services to end user organizations. But context granularity and fine tuning prior to the assessment could also be higher in this case.

Another challenge and potential problem when working with measurements and metrics is to define the difference between a good result, a bad result, and a neutral result. In the case of metrics related to release cycles in "Software Technology Attributes: Quality" in OpenBRR, they might be too rigid in the view of preferable release cycles. The same applies to QualOSS, when it comes to e.g. reporting of bugs and vulnerabilities. A trend indicating a rise in bug or vulnerability reporting has several potential interpretations, and all of them are not necessarily negative. Asterisk has experienced extreme growth in number of users the last couple of years. As a consequence, more functionality options have been explored and more hidden errors are found. This is one viable explanation. A challenge for assessment models like QualOSS and OpenBRR is not to punish more complex systems and systems with a large user community. Large projects with active communities will probably get many bug and vulnerability reports while a small project with very few users may get close to nothing. This does not in any way mean that the smaller project is more business ready or mature. The assessment results on bug and vulnerability reporting should be calibrated against the size of the user community, not only the developer community. A rising trend in reporting might indicate a rise in users, which is not necessary bad.

Both assessment approaches reacted on the high number of minor releases and patches of the Asterisk 1.4.x product line. Quite a high number of these minor releases and patches are produced to solve security issues based on reported vulnerabilities. This makes it in general more difficult to maintain a running Asterisk system from the perspective of a user organization and its system administrator. Both QualOSS and OpenBRR produce negative scores here as one could expect, but from the perspective of an Asterisk system administrator, the practical implications might not be that dramatic or time consuming. Apart from the core call processing functionality, which is establishing, maintaining, and ending connections, there are many options that can either be turned on or off at an Asterisk application. Therefore, each vulnerability alert has to be validated against the functionality of the running system to identify the need for maintenance.

Regarding documentation, OpenBRR gave a good score while QualOSS gave credit for documentation, but asked for more detailed design and system documentation to be satisfied. Taking a closer look at this finding, it is not possible to e.g. find a diagrammatic presentation of e.g. the core functionality of Asterisk. No design documentation is found either at least not for Asterisk 1.4. In the case of Asterisk 1.6, online reference documentation for Asterisk version 1.6.1.6 is available[2], but the quality of this has not been analysed any further here.

The most critical output of the QualOSS assessment was the lack of a holistic and structured test regime. This seems to be right at the time of the assessment in November 2009, but there is also a risk that some of the test results have either not been found or have not been made public. The latter may be true for a set of interoperability tests between Asterisk and e.g. other SIP based devices[3]. There is also a gathering called The SIPit, or Session Initiation Protocol Interoperability Test, that is a week-long event where various SIP implementations are assembled to ensure they work together. There is also confidentiality regarding interoperability test results from this event. Regarding performance testing some information is available, e.g. from third parties, but the information is not extensive. About two months after the QualOSS evaluation was performed Digium announced[4] increased focus on an Asterisk test framework consisting of the following components:

- Peer reviews
- Unit testing,
    - Through a new API in Asterisk trunk for writing unit tests within the code.
- An external test suite
    - Is about to be created
- Regression testing in combination with continuous code integration
    - Using Bamboo

[2] http://www.asterisk.org/docs
[3] This according to a person close to the core development team.
[4] http://lists.digium.com/pipermail/asterisk-dev/2010-February/042387.html

This is a clear indication that QualOSS did identify something that was really missing at the time of the assessment.

As explained in the background section, the reported OpenBRR activities are low and community is inactive. This is unfortunate since it seem to be a useful tool with small resource requirements. It needs general knowledge about where to find information on the Internet combined with deep domain knowledge on the part covering functionality.

OpenBRR should ideally have an active community that works with the models and legitimizes it. The risk of basing the whole assessment on manual work is that critical information can be missed. This is also the case for QualOSS, especially in the cases where no suitable tools are present. Then the options are either to perform the assessment on a manual basis or to do the assessment without full coverage of topics. Since the metrics and measurements are more complex than for OpenBRR the last option might sometimes be the right one. Whenever the tool support is working as intended the QualOSS is a source of more insight compared to a method like OpenBRR, as illustrated in some of the results presented in this article. On the risky side of QualOSS is the pre-determined definition of what is a good trend or score and what is not.

## V. Comparing the Assesment Methods

Generally, advantages of quality models are that they provide objective measurability and are simpler to use than more qualitative approaches. Disadvantages of quality models include a reduced notion of quality to a few relatively simple and static attributes, where they represent leaner and narrower perspectives on quality. These arguments also apply to OpenBRR and QualOSS. What characterizes QualOSS, as the current most powerful representative of a second generation FLOSS quality model, as compared to OpenBRR, representing the first generation of FLOSS quality models?

QualOSS and OpenBRR both cover several different views of quality, (i) the product view on quality, (ii) the manufacturing, or process view on quality, and also to some smaller extent (iii) the user view on quality. Similarities and differences between the two approaches are discussed below.

Both OpenBRR and QualOSS are defining their context and assessment scope based on business cases and the need of organizations wanting to assess FLOSS software. In QualOSS interviews happened before and after the QualOSS assessment in order ensure that the assessment methodology captured the relevant items, and to check if the results of the highly automated QualOSS assessment are good and understandable enough to convince people with expertise knowledge of the FLOSS endeavours under scrutiny.

When the scope is defined, QualOSS has a large set of predefined metrics and indicators based on GQM, the Goal Question Metrics approach. OpenBRR has a much smaller metrics set, containing 27 different metrics, which are predefined like for QualOSS. However, flexibility arises in OpenBRR when defining the feature set for the Functionality category, both in choosing the actual features (whether to include them as standard or extra), and setting their importance (1-3). This involves human experts into the OpenBRR process, and this type of interaction is not present in the QualOSS assessment, where detailed metrics (e.g. involving coding standards) are defined (at least for some programming languages).

OpenBRR defines seven quality characteristics (twelve in the first version), each having a mostly unique metrics set, except for two metrics that are used twice (“Average number of messages on mailing list” and “Reference deployments”). In QualOSS the metrics are representing various indicators for each of the quality characteristic leaf nodes like e.g., “Maintainability”, these are then further grouped into one of the “Work Products”, “Community”, and “Software Process” super categories, leading into the top node, “Robustness and Evolvability of the Endeavour”.

While The QualOSS assessment is a highly automated measurement and uses a number of measurement tools, OpenBRR is based solely on the skill of the evaluators. There is also a difference in the output of the two quality assessment models: while OpenBRR outputs a score, QualOSS also outputs trend indications (e.g. evolution of the number of lines of code between releases).

Related to the controversy of manual versus automated measuring, another difference between the first and the second generation quality model is striking: OpenBRR only allows assessment of a limited set of quality characteristics, and the use and popularity of the methodology shows a significant decline. QualOSS, in contrast, involves hundreds of quality metrics, and is designed to capture even more in future releases.

Time will show if QualOSS will succeed where the first generation models did not. The growth of a community around the assessment method itself seems to be a critical point.

## VI. Conclusions

What can be concluded from the evolution of quality models and the observed differences between the two quality models with regard to requirements from the advancement of FLOSS quality models? Does QualOSS, as a second generation quality model, outperform a first generation quality model like OpenBRR, or do both quality models complement each other?

The key trend in the evolution of FLOSS quality models is the movement from manual and descriptive to more automated and analytical models, and from the involvement of a few metrics to hundreds of metrics.

The question whether or not the second generation quality model can outperform the first generation model can only be answered with ambiguity. Both quality models have different strengths and weaknesses. OpenBRR has its strengths with regard to the concrete business case and the direct involvement of expertise knowledge in the interpretation of the assessment results. The weaknesses of OpenBRR are that this approach makes it vulnerable to subjective biases, and that the relatively limited number of metrics that can be examined increases the likelihood of missing important quality issues.

The strengths of QualOSS are surely its relative independence from direct individual influence on the measurement process and results and the huge number of metrics that is captured by the assessment. This feature makes it relatively unlikely that the QualOSS quality model misses important aspects of quality, as the above discussion of the QualOSS results with regard to testing illustrates. Its key weakness is a certain degree of opaqueness due to its complexity at the level of the detailed measures it is based upon.

Whenever the tool support is working as intended, the QualOSS is a source of more insight compared to a method like OpenBRR, as illustrated in some of the results presented in this article. On the risky side of QualOSS is the pre-determined definition of what is a good trend or score constitutes.

Overall, it appears that human expertise, especially knowledge of context conditions and development trends with a FLOSS endeavour, is decisive for the usability of both quality models. OpenBRR relies on this input by design. QualOSS tried to largely eliminate such direct input on the measurement process but, occasionally, seems to rely on it when tools are not available or when the results of the assessment must be interpreted. This is at least implied when the discussion above of possible different causes and meanings of increasing bugs and patches is considered.

Tht quality is a feature that is extremely hard to measure holds both for first generation FLOSS quality models as well as for second generation FLOSS quality models. Nevertheless, it appears that there is significant progress between the first generation and the second generation FLOSS quality model. However, it cannot be concluded that the first generation models with their relatively simplistic approach is outperformed by the second generation quality models. Still, human expertise, which is considered to be a weakness of first generation models, is significant for securing the reliability and validity of assessment results of second generation models.

Future FLOSS quality models therefore call either for an even stronger reliance on tools support, whereby the predetermination of the results as good, bad, or neutral must be minimized; or for an integration of the human factor and further efforts to minimize the subjectivity that is incorporated by doing so. As long as this has not been achieved, there is a need for both quality models because they complement each other, and they can in principle co-exist in the market. Therefore, OpenBRR should ideally have an active community working with the models and legitimizing it. Unfortunately, the reported OpenBRR activities are low and the community inactive. This is disappointing as it seems to be potentially a useful tool with small resource requirements. Similarly, the community support for QualOSS has still not reached its full potential, and there is scope to further develop this methodology. Time will show if an active community will grow around QualOSS or be regenerated around OpenBRR, or if another quality model will appear. It is at in any case clear that there is a real need for sound quality models in the market, helping actors make their decisions.

## ACKNOWLEDGMENT

We would like thank Dr. Wasserman from the Center for Open Source Investigation at Carnegie Mellon West for providing the updated OpenBRR spreadsheet.

## REFERENCES

[1] Garvin, D. A., "What does 'Product Quality' really mean?", Sloan Management Review, no. 1, pp. 25-43, 1984.

[2] Kitchenham, B. and Pfleeger, S. L., "Software quality: the elusive target [special issues section]", IEEE Software, no. 1, pp. 12-21, 1996.

[3] Crosby, P. B., Quality is free: the art of making quality certain, New York : McGraw-Hill, 1979.

[4] Deming, W. E., Out of the crisis: quality, productivity and competitive position, Cambridge Univ. Press, 1988.

[5] Huggins, L. P., Total quality management and the contributions of A.V. Feigenbaum, Journal of Management History, Vol 4, 1, pp. 60-67,1998

[6] Weinberg, G. M., "Quality Software Management, Vol. 3: Congruent Action" Dorset House Publishing Company, Inc., September 1994.

[7] Robson, C., Real world research: a resource for social scientists and practitioner-researchers, Blackwell Publisher Ltd., 2002.

[8] Paulk, Mark C., Weber, Charles V., Garcia, Suzanne M., Chrissis, Mary Beth, and Bush, Marilyn, "Capability Maturity Model for Software, Version 1.1", Software Engineering Institute, Carnegie Mellon University, 1993.

[9] Curtis, Bill, Hefley, Bill, and Miller, Sally, "People Capability Maturity Model® (P-CMM®), Version 2.0", Software Engineering Institute, Carnegie Mellon University, 2001.

[10] van Loon, H. Process Assessment and ISO 15504, Springer, 2007.

[11] ISO, International Organization for Standardization, "ISO 9001:2000, Quality management systems – Requirements", 2000.

[12] McCall, J. A., Richards, P. K., and Walters, G. F., "Factors in Software Quality", Nat'l Tech.Information Service, Vol. 1, 2, and 3, 1977.

[13] Boehm, Barry W., Brown, J. R, and Lipow, M.: Quantitative evaluation of software quality, International Conference on Software Engineering, Proceedings of the 2nd international conference on Software engineering, 1976.

[14] Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., and Merritt, M.: Characteristics of Software Quality, North Holland, 1978.

[15] ISO, International Organization for Standardization, "ISO 9126-1:2001, Software engineering – Product quality, Part 1: Quality model", 2001.

[16] Capgemini's Open Source Maturity Model (OSMM) assessment available at: http://www.osspartner.com/portail/sections/accueil-public/evaluation-osmm

[17] Wijnen - Meijer, M. Detailed comparison of existing open source software evaluation models, OSOSS, 2006 http://noiv.nl/files/2009/12/models_comparison_-_1_5.pdf

[18] Wilson, J. Open Source Maturity Model, 2006, http://www.oss-watch.ac.uk/resources/osmm.xml, section 3.

[19] OSS Watch at the University of Oxford also considers methods like OpenBBR to be useful. See: http://www.oss-watch.ac.uk/resources/brr.xml, section 4.

[20] Sclater, N. Enhancing & Embedding a Mission-Critical Open Source Virtual Learning Environment, 2006, http://www.oss-watch.ac.uk/events/2006-04-10-12/presentations/niallsclater.pdf

[21] Qualipso, Roadmap: OMM overview, http://qualipso.icmc.usp.br/OMM/

[22] Qualipso, CMM-like model for OSS, http://www.qualipso.org/node/175

[23] Samoladas I., Gousios G., Spinellis D., and Stamelos I. The SQO-OSS quality model: Measurement based open source software evaluation. In E. Damiani and G. Succi, ed., Open Source Development, Communities and Quality: 4th International Conference on Open Source Systems, pp. 237–248, Boston, September 2008.

[24] SpikeSource, Carnegie Mellon West, Intel, Business Readiness Rating for Open Source, 2005, http://www.openbrr.org/wiki/images/d/da/BRR_whitepaper_2005RFC1.pdf

[25] Deprez J.-C., Haaland K., and Kamseu F., QualOSS Methodology & QUALOSS assessment methods. QualOSS Deliverable D4.1. http://www.qualoss.org/about/Progress/deliverables/WP4_Deliverable4.1_submitted.pdf 2008

[26] Ruiz J., Glott R., Flamand J., Results of Case Studies. QualOSS Deliverable D5.3. http://www.qualoss.org/deliverables/qualoss%20test2.rtf . 2009

[27] Glott R., Haaland K., Ghosh R., and Deprez J.-C., Validation of Data and Measurements of Advanced F/OSS Projects. QualOSS Deliverable D3.3. http://www.qualoss.org/deliverables/WP3-D3.3_final_submitted.pdf . 2009

[28] Basili, 1992. V.R. Basili, Software Modeling and Measurement: The Goal/Question/Metric Paradigm. University of Maryland Technical Report. UMIACS-TR-92-96