Towards Automated Quality Models for Software Development Communities: the QualOSS and FLOSSMetrics case

Daniel Izquierdo-Cortazar, Jesus M. Gonzalez-Barahona, Santiago Dueñas, Gregorio Robles
GSyC/LibreSoft
Universidad Rey Juan Carlos
Mostoles, Spain
Email: {dizquierdo,jgb,sduenas,grex}@libresoft.es

***Abstract*—Quality models for software products and processes help both to developers and users to better understand their characteristics. In the specific case of libre (free, open source) software, the availability of a mature and reliable development community is an important factor to be considered, since in most cases both the evolvability and future fitness of the product depends on it. Up to now, most of the quality models for communities have been based on the manual examination by experts, which is time-consuming, generally inconsistent and often error-prone. In this paper, we propose a methodology, and some examples of how it works in practice, of how a quality model for development communities can be automated. The quality model used is a part of the QualOSS quality model, while the metrics are those collected by the FLOSSMetrics project.**
Keywords: Quality models automation, data mining, libre software, software metrics.

I. Introduction

During the last 40 years, many quality models for software and software development have been proposed. Usually, they focus on estimating the quality of the development process, the resulting product, or both. With the advent of FLOSS (free, libre, open source software)[1], the importance of estimating the quality of the development community supporting the project has become of special interest as well. As a result, it has been included with different levels of detail in most quality models for libre software.

However, up to now those models addressing the quality of development communities have been based on manual or semi-manual estimation by experts, which is, among other issues, time-consuming, error prone, inconsistent among projects and experts, and subject to the availability of those experts. In other words, up to now several quality models have been proposed, but in most of the cases the way to retrieve the data and calculate the metrics has not been defined (by a methodology and by a set of tools) in a strict sense, and each expert interested in using a specific quality model has to use her own tools. In many cases, tools simply do not exist and the retrieval process has to be done manually (which may produce inconsistent results).

Our approach to avoid those problems is to fully automate the process, thanks to the availability of data repositories with detailed information about how software is being developed. These repositories are, in the case of libre software, common and publicly available. The approach allows for the transparent, reproducible, and affordable continuous evaluation of quality models, showing how the quality parameters of the communities evolve over time.

In this paper we discuss how this automation is actually done in the case of a specific quality model, the QualOSS model [1], specially targeted to libre software. The methodology discussed is based on using the data obtained from software development repositories by the FLOSSMetrics project[2]. Besides, this process could be used in other quality models, where objective metrics have been defined.

II. State of the Art

Many software quality models have been proposed, since the classical industry-oriented studies by McCall [2] and Boehm [3]. Of those, some of the most remarkable among the recent ones are the quality model of the Deutsche Gesellschaft für Qualität [4], the NASA SATC [5], and the ISO 9126 standard [6].

There have been many discussions about how an extension of these models can be applied to libre software projects, since many of the assumptions common in traditional, industry-based projects are not found in the libre software world. In particular, the importance of a development community that usually includes a mixture of voluntary and hired developers, working with different intensities, goals and motivations causes completely different scenarios. Developers range from core developers, that take most of the decisions, contribute most of the effort, and produce most of the code, to casual bug reporters or mailing-list participants [7].

In order to consider the peculiarities of libre software development, several quality models have been specifically

[1]In this paper, we will use the term "libre software" to refer both to "free software", as defined by the Free Software Foundation, and "open source software", as defined by the Open Source Initiative. In some cases, we will also use FLOSS (free, libre, open source software).

[2]http://flossmetrics.org

been proposed, such as OpenBRR[3] or QSoS[4]. They are based on a set of metrics of different kinds, related to parameters of the product, the development process, and the supporting community. Their aim is to produce a final characterization (via a mark or set of marks) to show the readiness of the software for general (industrial, corporate, government, etc.) use. Those models are not supported by automatic tool-chains, which means that applying them is a tedious and time-intensive task [8].

In fact, both the OpenBRR and QSoS quality models do not present a way to automatize the metrics, and more specifically the community side. In the following, the QualOSS[5] method and the SQO-OSS[6] quality model are presented, as they try to provide automated means to gather community-related information.

QualOSS (Quality of Open Source Software) was designed as a quality model with the goal of allowing the comparison of libre software products in an objective, semi-automated, simple and fast way. Robustness and evolvability of libre software endeavors are key aspects for software companies that increasingly integrate FLOSS components in their solutions and systems to produce quickly new, reliable applications and services. QualOSS considers a FLOSS endeavor to be composed of a set of community members (or contributors), a set of work products including code, a set of development processes followed by the community to produce work products, and a set of tools used to support the endeavor, to produce work products and to run the FLOSS software component [9].

The SQO-OSS project was also designed as a quality model with the goal of automation and as a continuous quality monitoring system [10]. Its main quality attributes are both product quality and community quality. Compared to QUALOSS, there are fewer metrics regarding the analysis of the community.

This paper will be devoted to applying the QUALOSS method, as we have in it the most expertise. However, other quality models such as SQO-OSS, OpenBRR or QSoS are presented and calculated their level of automation following the proposed methodology. Hence, this paper does not try to compare these methodologies.

III. Methodology

The methodology proposed consists of the study of a given quality model related to open source and check the quality attributes which are matched with a set of metrics. In most of the cases (like in the case of OpenBRR or QSoS and mostly in the case of SQO-OSS or QualOSS) the metrics are well defined, and are objective ones, in the sense that for a given metric there is no doubt about how to calculate it.

Thus, for each quality attribute, there are several metrics, which later are combined to provide a final value. These quality attributes are again aggregated to conform the final value of the quality attribute which aims to provide information about the current status of the project.

All of the aforementioned quality models related to the libre software world, provide a set of metrics related to community and some of them could be fully or partially automated. Thanks to the publicly available data sources, the FLOSSMetrics project offers a way to avoid all the retrieval process, adding value in terms of traceability and trustworthiness of results.

The methodology proposed is based on the public access to data sources, that facilitates the retrieval analysis process. This means that projects have to offer those data sources in order to work with this.

The followed methodology is based on next steps:

- All the quality models focused on libre software projects provide a set of metrics. Since we are focusing this analysis in measuring quality of communities, those are the metrics which should be taken into account. However, this study could be extended to all the metrics.
- Secondly, for each of the selected metrics, it is necessary to detect the associated data sources. In sub-section III-A, some of them will be briefly described being the most important the source code management systems, the mailing lists, the bug tracking systems and the releases of source code. Thus, each metric could be matched with more than one data source.
- As a third step, a way to manually calculate the metric is necessary. This is crucial if there are not possibilities to be automatically retrieved.
- Finally, and taken into account the FLOSSMetrics database schema, the generation of the query has to be done. The FLOSSMetrics database offers information from the main repositories mentioned above and a study about how to automate the manual analysis is done. This step will outcome a set of manual, semi-manual or fully automated metrics.

A. Sources of data

The methodology for automation discussed in this paper is mainly focused on the development community aspects of version 1.0.1 of the QualOSS specification [11], although this is also applied, at least in part, to some other quality methods, such as OpenBRR, QSoS or SQO-OSS.

Since the QualOSS evaluation is meant to be reproducible, it is usually based on the availability of public data about the projects to evaluate. In the case of FLOSS projects this is usually the case, since most of them offer all the information publicly in the software development repositories. For proprietary systems or projects without a public available data source, the analysis could only be done by the project managers or those third parties with access to the data.

The repositories providing information to feed the QualOSS model are:

- Source code management repository. Used to maintain versions of all changes to the source code, along with

[3]http://www.openbrr.org
[4]http://www.qsos.org
[5]http://qualoss.eu
[6]http://www.sqo-oss.org

meta information, such as who and when performed the changes.
- Mailing lists repository. It stores messages exchanged by developers and other members of the project community, and constitutes the main asynchronous communication method of the project.
- Issue tracking system. Stores tickets for all reported issues, including bug reports. Reports are usually submitted when a bug is detected, but not only then.
- Source code releases repository. At specific points in time, projects usually produce a release of the source code tagging them according to the maturity of the release as alpha, beta, stable, etc.

Our approach is based on the data structure and tools used by FLOSSMetrics [12], a project that has analyzed, from a quantitative point of view, thousands of FLOSS projects. To address the complexity and the difficulties that can be found during the analysis of a large amount of data sets [13], FLOSSMetrics has developed a platform that solves most of them and automates the data retrieval and basic analysis. This platform downloads information from FLOSS project repositories, and stores it in a database, which is later further analyzed by a set of specific tools. The database dumps are publicly available via the Melquiades system[7].

B. FLOSSMetrics

The structure of the FLOSSMetrics database has been designed to cover the various needs of researchers and software developers. It is divided in several layers, focused to the various studies to be performed. The first layer contains the data directly as extracted by the retrieval tools; the second layer unifies the data of the previous one, based on several criteria; and the third layer contains analysis and statistics.

Specifically, the QualOSS model is based on the data provided by the third layer of the database. Several scripts, composed in part of SQL queries, have been developed to obtain the final marks when applying the quality model to a given project.

All the databases offered by the FLOSSMetrics project can be found at the Melquiades website [8]. Thus, if we are interested in a given project, this is enough to look for it using the searching box available. This will show a new web site with a set of dumps ordered by date and by type of data source.

IV. The QualOSS model

The QualOSS model is based on a GQM (goal-question-metric) methodology, from which relevant goals, questions, and finally metrics and indicators were derived [14]. The computation of metrics measurements and aggregation for answering questions of the QualOSS quality model were partially automated with several tools. As was already stated, in this paper we focus on the aspects related to the analysis of the communities supporting projects.

[7]http://melquiades.flossmetrics.org
[8]http://melquiades.flossmetrics.org/

The QualOSS quality model version 1.0.1 [11] defines three main quality attributes for communities: Size and regeneration adequacy, Interactivity and workload adequacy, and Composition adequacy. For each of the quality attributes, several metrics were defined to obtain objective answers. As a result, there are a total of 29 metrics, with 11 of them corresponding to the first quality attribute, 11 to the second one, and 7 to the last attribute.

After a careful analysis, 22 out of the 29 metrics were found to be are fully automatize, if the data sources available from each project were appropriate. From these 22 metrics, there are 15 which are retrieved from a source code management system, 4 from a bug tracking system analysis and 3 from the mailing lists corresponding to the analyzed project.

When a metric cannot be calculated because its data source is not available, the QualOSS method just ignores it, adding a control flag designed to quantify the 'confidence' of the quality attributes (which will be lower when more metrics were ignored).

A. Quality Attributes and Metrics

In this subsection, the list of metrics related to the community aspects of FLOSS endeavors are listed. Those related to size and regeneration adequacy are:

- CM-SRA-1: Evolution of first bug submitted by registered people
- CM-SRA-2: Evolution of first commit submitted by registered people
- CM-SRA-3-SCM: Evolution of first no source code commit by registered people
- CM-SRA-3-BTS: Evolution of first no bug report submitted by registered people
- CM-SRA-3-MLS: Evolution of first post in mailing lists
- CM-SRA-4: Evolution of new core contributors
- CM-SRA-5: Evolution of core member leaving core team
- CM-SRA-6: Evolution of balance in the core team
- CM-SRA-7: Average commiters longevity
- CM-SRA-8: Evolution of code contributors who submitted patches and changes
- CM-SRA-9: Total code contributors who submitted patches and changes

Those related to interactivity and workload adequacy:

- CM-IWA-1-SCM: Evolution of the number of events (in SCM)
- CM-IWA-1-BTS: Evolution of the number of events (in issue tracking system)
- CM-IWA-1-MLS: Evolution number of events (in mailing lists)
- CM-IWA-2: Evolution of the number of commits
- CM-IWA-3: Percentage of people working with old releases
- CM-IWA-4: Territoriality
- CM-IWA-5-SCM: Lines per committer
- CM-IWA-5-BTS: Bugs per committer
- CM-IWA-5-MLS: Emails per committer

- CM-IWA-6: Lines per committer
- CM-IWA-7: Percentage of the number of handled files per committer

Finally, those which are based on the quality attribute composition adequacy are

- CM-CA-1: Companies reporting bugs
- CM-CA-2: Companies committing
- CM-CA-3: Companies with community leaders
- CM-CA-4: Companies committing in module
- CM-CA-5: Roles filled by community members
- CM-CA-6: Are roles adequate
- CM-CA-7: Is code expert enough

B. Color codes for results

The QualOSS model shows as results, from a business perspective the risks of working with a certain FLOSS community. For a summary of those results, a color code of four colors was defined, from higher to lower risk: black, red, yellow and green. Those colors are indicators, not to be trusted blindly. In particular, they should not be considered separately, but together with the rest of quality attributes to form an overall risk picture associated to a project.

The values used as thresholds for choosing the colors (once the related metrics are calculated) are based on the results from a benchmarking study [14] which used a set of around 1400 projects from the FLOSSMetrics project. Therefore, a project for which most of the final indicators are black or close to black, could be interpreted a that the community of that project has shown in that benchmarking sample a risky situation.

V. Metrics and Indicators

Since our aim is to show how the metrics can be calculated directly from FLOSSMetrics data, in this section it will be shown in detail, as an illustration, how three of them are obtained automatically. For each of those metrics, the following information is provided: detailed description, the rationale or methodology applied and the indicators with the thresholds as defined by the QualOSS method. Some of the queries used in the QualOSS method can be found at the Melquiades wiki [9].

A. Average of committers longevity

- *Description*: This metric is focused on obtaining the average number of months that developers have spent contributing to the project, offering a measure of the *seniority* of the development team.
- *Methodology*: For each committer in the project, obtain the number of months in which at least one commit by her is found. This is considered as her age in the project. The average of all of them is the average longevity of committers in the community.
- *Indicator thresholds*:
  - *Black*: $[1, 5.53555]$
  - *Red*: $]5.53555, 8.5789]$
  - *Yellow*: $]8.5789, 12.25]$
  - *Green*: $]12.25, +\infty[$

[9]http://melquiades.flossmetrics.org/wiki/doku.php

Metric	Ant	HTTPD 1.3.x	Eclipse	Nautilus
cm-sra-1	Green	Red	Green	Green
cm-sra-2	Red	Red	Red	Red
cm-sra-3	Yellow	Yellow	Yellow	Green
cm-sra-4	Red	Red	Red	Green
cm-sra-5	Black	Black	Black	Yellow
cm-sra-6	Red	Red	Red	Red
cm-sra-7	Green	Green	Green	Green
cm-sra-8	Yellow	Yellow	Green	Yellow
cm-sra-9	Red	Red	Green	Red
cm-iwa-1	Yellow	Yellow	Yellow	Yellow
cm-iwa-2	Red	Red	Red	Red
cm-iwa-3	Red	Red	Red	Red
cm-iwa-4	Yellow	Green	Yellow	Green
cm-iwa-5	Yellow	Green	Red	Red
cm-iwa-6	Green	Green	Red	Green
cm-iwa-7	Green	Green	Green	Green

TABLE I
QualOSS community indicators for Ant, HTTPD 1.3.x, Eclipse and Nautilus

B. Territoriality

- *Description*: Indicates how many files are worked on by a single committer. Most of the files are "touched" or handled by just one committer, which means that there are high levels of territoriality. This situation may be seen as a risky one as if the developer leaves the project, the maintainability of the source code will suffer.
- *Methodology*: Percentage of files worked on by just one committer.
- *Indicator thresholds*:
  - *Black*: $]87.245, +\infty[$
  - *Red*: $]71.24, 87.245]$
  - *Yellow*: $]55.22, 71.24]$
  - *Green*: $[0, 55.22]$

VI. Case studies

As a final illustration of the feasibility and usefulness of this automated approach to the obtaining of quality indicators for the community supporting the projects, the results of applying it to a number of FLOSS projects is offered in Table I. All these results are available, among many others, at the QualOSS website[10]. All the results, except some minor parts, were automatically retrieved and calculated using the database dumps available from FLOSSMetrics.

VII. Other Quality Models

The QualOSS section aimed to focus on a specific quality method in order to show how the methodology was applied, however, there are other quality models which were mentioned.

Checking the quality models given by the OpenBRR, QSoS and SQO-OSS method, they also threat the community side

[10]http://ingrid.cetic.be:33323/qualoss_assessment/

as a key factor to be taken into account. This is carried on by means of a specific quality attribute named as community or through the definition of several metrics related to this.

Thus, this section aims to show that the methodology is also applicable to other quality methods and not only the one proposed by the authors.

Since the goal of this paper is not to present a state of the art in quality models, but to show a way to automatize all of the quality models by means of the use of the available data in meta-repositories such as FLOSSMetrics the next subsections will not deeply explain each of the quality models. However, we will show how the methodology is applied in other quality methods different than the QualOSS method.

Briefly, the methodology consists of the next steps:

- Take all the metrics related to community.
- For each of the detected metrics, check its primary data source.
- Create a manual way to calculate that value.
- Create an automated way to calculate that value using the data available in FLOSSMetrics.

A. OpenBRR

OpenBRR is used as a model to rate libre software projects. This quality assessment method shows an initial set of metrics related to community, but the authors foster the extension of this by others.

In the specific case of OpenBRR, there is a quality attribute named as Community. This provides two metrics with next characteristics (let us name them as OpenBRR-1 and OpenBRR-2):

- OpenBRR-1: Average volume of general mailing list in the last 6 months.
  - *Scoring (messages per month)*:
    - *5-Excellent*: $]+\infty, 720]$
    - *4-Very Good*: $]720, 300]$
    - *3-Acceptable*: $]300, 150]$
    - *2-Poor*: $[150, 30]$
    - *1-Unacceptable*: $[30, 0]$
- OpenBRR-2: Number of unique code contributors in the last 6 months.
  - *Scoring (unique code contributors)*:
    - *5-Excellent*: $]+\infty, 50]$
    - *4-Very Good*: $]50, 20]$
    - *3-Acceptable*: $]20, 10]$
    - *2-Poor*: $[10, 5]$
    - *1-Unacceptable*: $[5, 0]$

Both metrics are fully automatize by using the FLOSSMetrics database and checking that the projects fit with the data sources available in this project. As explain above, there is a specific database for the mailing lists and another mailing list with data from contributors to the source code.

B. QSoS

This methodology is close to the goals of OpenBRR, but they do not really identify the community as a quality attribute. However, checking the quality attributes provided by the QSoS method, there are at least two of them directly related to the community side inside the "Intrinsic durability" quality attribute:

- Development leadership:
  - Leading team:
    - *Scoring (individuals involved)*:
    - *0-Poor*: $[1, 2]$
    - *1-Medium*: $]2, 5]$
    - *2-Good*: $]5, +\infty[$
  - Management style: This sub quality attribute does not provide a set of objective values to be measured.
- Activity:
  - Developers, identification turnover:
    - *Scoring (individuals)*:
    - *0-Poor*: $[0, 3]$
    - *1-Medium*: $[4, 7]$
    - *2-Good*: $[8, +\infty[$
  - Activity on bugs: No objective values provided.
  - Activity on functionalities: No objective values provided.
  - Activity on releases: No objective values provided.

Five metrics were identified as being directly related to community. One out of two from the quality attribute "Develoment leadership" could be automatized due to the fact that objective thresholds are provided. In the case of the quality attribute "Activity", only one out of four metrics could be automatized since the other three metrics do not provide an objective way to be calculated.

C. SQO-OSS

The SQO-OSS method proposed by the SQO-OSS consortium provides three sub quality attributes related to the community side. This projects offers a platform which retrieves all the data. Thus, this project has automatized the retrieval process for the quality attributes.

This method divides the community quality attributes in three main sections: mailing list, documentation and developer base:

- Mailing list:
  - Number of unique subscribers.
  - Number of messages in user/support list per month.
  - Number of messages in developers list per month.
  - Average thread depth.
- Documentation:
  - Available documentation documents
  - Update frequency
- Developer base:
  - Rate of developer intake
  - Rate of developer turnover
  - Growth in active developers

In this specific case, and after studying the definition of the metrics, those based on mailing lists and developers are

Quality Method	Num. metrics	Num. automat. metrics
QualOSS	29	22
SQO-OSS	9	7
OpenBRR	2	2
QSoS	6	2

TABLE II
QUALITY METHODS AND THEIR AUTOMATIZED COMMUNITY METRICS

fully automatized. On the other hand, the documentation metrics, since some of them are based on documentation, this information is partially provided by the aforementioned data sources what makes more difficult to calculate them.

All in all, seven out of 9 metrics of the SQO-OSS method related to the community side are fully automatized.

VIII. Conclusions and Further Work

In this paper, we have tried to show how the calculation of the indicators produced by a quality model can be automated, in order to easily retrieve the necessary data. Although the paper has focused on the quality of the community supporting a project, the method can be extended to other areas.

The discussed methodology has been presented for the QualOSS method, and is based in the availability of several software development repositories (SCM, issue tracking, mailing lists). The data corresponding to these repositories are either downloaded from the FLOSSMetrics database, or retrieved directly from them with the FLOSSMetrics retrieval and analysis tools. The paper has shown how some specific metrics can be calculated from the database, how the whole process can be followed for an specific project, and the results of applying the methodology to a set of projects.

Summarizing, the results to automatize as many as possible metrics from several quality methods focused on libre software are shown in table II.

However the methodology has shown some limitations:

- Data sources are not found. The proposed framework works with most of the well known libre software repositories. However, those are not all of them. This may cause situations where the data is not available. This is also a limitation of the quality method applied, which should take this fact into account.
- Metrics are too complex to be retrieved. So far, some of the metrics can not be retrieved, since the tools needed for that are still not available. This is the case, for example, of the "Composition Adequacy" indicator in the QualOSS model. Those metrics need to analyze code as published in releases, which is still not possible with the current set of tools found in FLOSSMetrics.
- The quality methods do not provide an objective way to calculate the metrics, specifically a set of thresholds which determine how critical is a given metric for the project. An example of this is the metric related to the "Activity on functionalities" found in the QSoS method.

The proposed methodology is not specific to QualOSS, although that was the wider example. Some other quality methods such as OpenBRR or QSoS have been shown that provide a set of metrics related to community.

In the case of QualOSS in particular, but also in other software development quality models, automated calculation of indicators can be also done in other areas, besides those related to the community. For instance, in the case of the metrics related to software development processes, some of them are based on the number of open or closed bug reports, or on the time to fix a bug report.

Acknowledgments

This work has been funded in part by the European Commission, under the QualOSS (FP6-IST-5-033547) and FLOSSMETRICS (FP6-IST-5-033547) projects, and by the Spanish CICYT, under SobreSalto project (TIN2007-66172).

References

[1] J.-C. Deprez, "Standard qualoss assessment method version 1.1," QualOSS Consortium, Tech. Rep., 2009. [Online]. Available: http://www.qualoss.org/deliverables/D4.5_StdQualOSSAssessmentMethod-v1.1.tar.bz2

[2] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in software quality," Tech. Rep., 1977.

[3] E. al. Barry W. Boehm, *Characteristics of Software Quality (TRW series of software technology)*. Elsevier.

[4] "Deutsche Gesellschaft fuer Qualitaet, Software-Qualitaetssicherung," VDE-Verlag Berlin, Tech. Rep., 1986.

[5] L. Hyatt and L. Rosenberg, "A software quality model and metrics for risk assessment," in *ESA'96 Product Assurance Symposium and Software Product ASS*.

[6] *ISO/IEC 9126 International Standard, Software engineering-Product quality, Part 1: Quality model*, 2001.

[7] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of Open Source software development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 3, pp. 309–346, 2002.

[8] J.-C. Deprez and S. Alexandre, "Comparing assessment methodologies for free/open source software: OpenBRR & QSOS (to appear)," in *Proceedings of the 9th International Conference on Product Focused Software Process Improvement (PROFES 2008)*, June 2008.

[9] J.-C. Deprez, F. Fleurial-Monfils, M. Ciolkowski, and M. Soto, "Defining software evolvability from a free/open-source software," in *Proceedings of the Third International IEEE Workshop on Software Evolvability*. IEEE Press, October 2007, pp. 29–35.

[10] I. Samoladas, G. Gousios, D. Spinellis, and I. Stamelos, "The SQO-OSS quality model: Measurement based open source software evaluation," in *Open Source Development, Communities and Quality — OSS 2008: 4th International Conference on Open Source Systems*, E. Damiani and G. Succi, Eds., IFIP 20th World Computer Congress, Working Group 2.3 on Open Source Software. Boston: Springer, 2008, pp. 237–248. [Online]. Available: http://www.dmst.aueb.gr/dds/pubs/conf/2008-OSS-qmodel/html/SGSS08.htm

[11] J.-C. Deprez, "Reference f/oss project report," QualOSS Consortium, Tech. Rep., 2008. [Online]. Available: http://www.qualoss.org/about/Progress/deliverables/WP4_Deliverable4.1_submitted.pdf

[12] I. Herraiz, D. Izquierdo-Cortazar, and F. Rivas-Hernández, "Flossmetrics: Free/libre/open source software metrics," in *CSMR*, 2009, pp. 281–284.

[13] G. Robles, J. M. Gonzalez-Barahona, D. Izquierdo-Cortazar, and I. Herraiz, "Tools for the study of the usual data sources found in libre software projects," *International Journal on Open Source Software and Processes*, vol. 1, no. 1, 2008.

[14] D. Izquierdo-Cortazar, J. M. Gonzalez-Barahona, G. Robles, J.-C. Deprez, and V. Auvray, "Floss communities: Analyzing evolvability and robustness from an industrial perspective," in *OSS*, 2010 (forthcoming).