FLOSS Communities: Analyzing Evolvability and Robustness from an Industrial Perspective*

Daniel Izquierdo-Cortazar[1], Jesus M. Gonzalez-Barahona[1], Gregorio Robles[1], Jean-Christophe Deprez[2], and Vincent Auvray[3]

[1] GSyC/LibreSoft, Universidad Rey Juan Carlos, Mostoles, Madrid
{dizquierdo,jgb,grex}@libresoft.es
[2] Centre of Excellence in Information and Communication Technologies, Charleroi, Belgium jean-christophe.deprez@cetic.be
[3] PEPITe, Liège, Belgium v.auvray@pepite.be

Abstract. Plenty of companies try to access Free/Libre/Open Source Software (FLOSS) products, but they find a lack of documentation and responsiveness from the libre software community. But not all of the communities have the same capacity to answer questions. Even more, most of these communities are driven by volunteers which in most of the cases work on their spare time. Thus, how active and reliable is a community and how can we measure their risks in terms of quality of the community is a main issue to be resolved. Trying to determine how a community runs and look for their weaknesses is a way to improve themselves and, also, a way to obtain trustworthiness from an enterprise point of view. In order to have a statistical basement, around 1400 FLOSS projects have been studied to create thresholds which will help to determine a project's current status compared with this initial set of FLOSS communities.

Keywords. Libre software communities, quality models, data mining.

1 Introduction

QualOSS [2] (Quality of Open Source Software) is a research project focused on the assessment of the quality of FLOSS (free, libre, open source software) endeavor. A FLOSS endeavor is 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

* This work has been funded in part by the European Commission, under the FLOSSMETRICS (FP6-IST-5-033547), QUALOSS (FP6-IST-5-033547) and QUALIPSO (FP6-IST-034763) projects, and by the Spanish CICyT, project SobreSalto (TIN2007-66172).

[2] The QualOSS project is coordinated by CETIC, and includes also University of Namur, Universidad Rey Juan Carlos, Fraunhofer IESE, Zea Partners, UNU-MERIT, AdaCore and PEPITe. The work described in this paper has been performed, or coordinated, mainly by the GSyC/LibreSoft group at Universidad Rey Juan Carlos. More info about the project: http://qualoss.org/

FLOSS software component [2]. In other words, a FLOSS endeavor is really like an enterprise working on FLOSS development projects. In turn, the exact goal of QualOSS aims at assessing the robustness and evolvability of FLOSS endeavors.

When acquiring software, enterprises are not only interested to know about the product and its quality but also interested in who produced that product and its reputability. For traditional enterprises, reputability can be check based on financial strength of the software provider. However, for the FLOSS world, we must find other ways to determine if a FLOSS endeavor (or a FLOSS project) is serious. This can be done by studying the behavior of a FLOSS community. In particular, a FLOSS community should behave in a manner to convince potential FLOSS integrators from industry that it is dependable.

2 Related Research

In the area of FLOSS, several models have been proposed as well, such as OpenBRR[3] or QSoS[4]. They consider metrics in several realms relevant to FLOSS development and maintenance, ranging from product to process or community metrics. In addition, some of them lack the needed benchmarking to fine-tune the methodologies proposed, and in some cases do not consider some important aspects of FLOSS development or maintenance [1]. Finally it is necessary to show that in the case of OpenBRR the number of metrics associated to the community side are just two metrics. On the other hand, QSoS provides four metrics related to activity over the source code.

QualOSS is aimed to fill this gap [6], and specifically in terms of community assess to provide a methodology whose metrics and indicators are semi-automatically retrieved and all based on a theoretical framework. Thus, results are influenced by the existing methodologies and their lack of information regarding communities, what we think that it is a key factor to take into account in software maintenance process, as well as interviews with FLOSS integrators.

3 Methodology

For achieving its aims, QualOSS started by applying a Goal-Question-Metric methodology, from which relevant goals, questions, and finally metrics and indicators were derived. The computation of metrics measurements and aggregation for answering questions of the QualOSS quality model was partially automated with several tools.

Interviews and GQM.

[3] `http://www.openbrr.org`
[4] `http://www.qsos.org`

The first step of the QualOSS methodology consisted of gathering the current state of the art on the topic, from a theoretical and empirical point of view [3]. In addition, some companies were interviewed to know about their needs, direct or indirect, regarding the quality of software. Those interviews were held with the goal of identifying the needs from an industrial point of view. Focusing on the community side, companies with a business model specifically-based on FLOSS are usually not directly worried to use products still in their preproduction state and with no stable releases. They highlighted the importance of the surrounding community and the support it may provide. Therefore these companies do not hesitate to interact with the community, sharing technical and non-technical goals

Using the Goal-Question-metric approach, a goal is defined by an issue, a context, a point of view and the object to analyze. In QualOSS, the issues consist of identifying the risk to collaborate with a community. The context assumes that an enterprise considers integrating a FLOSS component and collaborates fully with the existing FLOSS community. The point of view represents the role of people in the enterprise who are concerned about the community issue. And finally, the object to analyze is the FLOSS endeavor itself.

Then, to polish the quality focus it was necessary to create a quality model based on the ISO 9126 and merging it with the criteria proposed by the companies. From the information above, it is then possible to refine a set of goals related to community. For each goal, a set of questions determine how to verify if the goal is fulfill. Those questions are: First: "how can we measure community robustness?". Second: "how can we measure community evolvability?".

Both questions were further elaborated into several sub-questions, with the aim of characterizing the object of measurement.

Size and Regeneration Adequacy:

- **Definition**: The degree to which the size evolution and regeneration of a FLOSS community happens at an adequate rate to maintain a sustainable community size.
- **Questions**:
  - **sra2**- New code contributors evolution.
  - **sra3**- New non-code contributors evolution.
  - **sra4**- New core contributors evolution.
  - **sra5**- Evolution of core members who stopped contributing.
  - **sra6**- Balance between new core contributors and those who left the project.
  - **sra7**- Average longevity of committers to the FLOSS endeavor
  - **sra9**- Number of code contributors submitting changes in major releases.

Interactivity and Workload Adequacy:

- **Definition**: The degree to which the community interacts adequately and partition the workload among FLOSS community members adequately to maintain a community cohesion and motivation.
- **Questions:**

- **iwa1**- Is the number of events adequate (to show a lively community)?
- **iwa2**- Is the number of code commits adequate?
- **iwa4**- Are there sub-groups disconnected or are active community members serving as bridges between these sub groups?
- **iwa5**- Is there enough community supporting a FLOSS desired version?
- **iwa7**- Is the current team concerned about the entire source code?

It should be noticed that there are missing questions. Since this analysis is focused on the source code management system (the most usual data source found in FLOSSMetrics databases), those metrics related to other data sources (and so, their questions) have been removed.

Indicators.

For each metric defined above (each metric answers one question), there is a high-level risk indicator. This indicator measures one aspect of the risk taken by a company engaging in a full FLOSS collaboration. The QualOSS project has defined the following four color-coded risk levels, in order of decreasing risk: black, red, yellow and green. However, it should be noticed that indicators should not be trusted blindly. In particular, they should not be considered separately, but rather jointly to form an overall risk picture associated to a project.

In this paper, we adopt a data-driven approach to define indicators. Using the notion of quantile, we search for a partition of a metric's values into a given number of intervals with equal probability.

To compute our metrics and estimate our indicators, we use data collected by the FLOSSMetrics [5][5] project. Many of the open-source projects considered by FLOSSMetrics appear to be very small and are thus not representative of our population of interest. Indeed, we are assessing the risks associated to a full FLOSS collaboration with projects. This precludes very small projects for which, from a business perspective, a fork should be more appropriate.

4 Threats to Validity

As it was said, during the extraction data from the FLOSSMetrics databases, it did not provide full of data regarding other data sources except for the source code management system. Thus, what it is presented in this paper is just based on those metrics which are retrieved specifically from that data source.

It is also necessary to deal with the fact that projects stored in FLOSSMetrics database do not represent the whole population of FLOSS projects.

Finally, depending on the policy, projects may have a small set of developers who are in charge of committing all the changes. This will skew the results produced by our methodology.

[5] `http://flossmetrics.org`

5 Results for Illustration

In first place, an approach using the slope as our metric and then defining the indicators showed a lack of information since most of the slopes were pretty similar. We used another approach based on the slope, but modifying the way the indicator is calculated. In fact, those indicators are not dependable of a given statistical approach and some other may be used to improve the indicator accuracy. Projects presented for illustration are well known projects and even when the Evince community is more active than the Nautilus's or HTTPD1.3's, all of them show a similar activity in terms of commits per committer, handled files and other metrics.

Pred. diff.	evolution	evince	nautilus	httpd1.3
sra7	8.6044 Y	5.0272 B	7.4484 R	15.7206 G
iwa4	0.3666 G	0.4216 G	0.3168 G	0.0923 G
iwa5	374772.4 B	48282.8 R	246991.9 B	-
iwa7	0.145 G	0.1904 Y	0.0491 Y	6.0e-4 B
Δ_{90} sra2$_{90}$	0 Y	0 Y	0 Y	0 Y
Δ_{90} sra3$_{90}$	0 Y	0 Y	0 Y	0 Y
Δ_{90} sra9$_{90}$	0 Y	0 Y	0 Y	0 Y
Δ_{90} iwa1$_{90}$	-0.23327969 R	-0.0312513 R	-0.20157016 R	-0.62113349 R
Δ_{90} iwa2$_{90}$	-1.03448107 R	0.31998312 Y	-1.60891953 R	-0.76343814 R
Δ sra4	1 G	2 G	0 Y	0 Y
Δ sra5	1 B	1 B	2 B	0 R
Δ sra6	-1 R	0 Y	-1 R	0 Y

Table 1: Illustration in some projects using a linear model

6 Conclusions and Further Work

We have presented a way to estimate "quality" from an industrial perspective based on statistical analysis of hundreds of FLOSS projects. FLOSS communities are key actors in the software evolution and maintenance process and better understanding their behavior through their life will improve the make decision process.

For instance, checking the tendency by means of the methodology explained in this paper, we are able to know if a community is growing, or if the number of core committers is decreasing over and over. Perhaps we are interested in a specific product and we know that some of its weaknesses are motivated because of a really high turnover of developers.

Thus, we can check how reliable are the FLOSS communities looking at the values for the given set of metrics. As it was mentioned, indicators define

relative, and not absolute, risks. A low risk does not mean that the metric value is good, this is just good compared to other projects. In this case, this is useful if we are interested in guessing the activity of the community, the general tendency and how it behaves compared to some other set of projects.

As further work, we should say that indicators must be polished by using new data from FLOSSMetrics (current status of the Melquiades database [6] shows an increase of 600 projects since results were retrieved for this paper). There are other publicly available data sources which may be checked in order to add more accuracy to this analysis. Specifically OSSMole [7] or Ohloh [8] are some examples which have been used for academic purposes. The FLOSSMetrics project is currently adding data from bug tracking systems what means that some other indicators will be added using the statistical approach defined here.

We also need to include advanced aspects of community behaviors and how they can be integrated in the QualOSS methodology. For instance, community-driven projects show interesting interactions among participants [8, 4, 7].

References

1. 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.
2. 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*, pages 29–35. IEEE Press, October 2007.
3. J.-C. Deprez, J. Ruiz, and I. Herraiz. Evaluation report on existing tools and existing f/oss repositories. Technical report, QualOSS Consortium, 2007.
4. J. M. González-Barahona, L. López-Fernández, and G. Robles. Community structure of modules in the apache project. In *Proceedings of the 4th Workshop on Open Source Software Engineering*, Edinburg, Scotland, UK, 2004.
5. I. Herraiz, D. Izquierdo-Cortazar, and F. Rivas-Hernández. Flossmetrics: Free/libre/open source software metrics. In *CSMR*, pages 281–284, 2009.
6. D. Izquierdo-Cortazar, G. Robles, J. M. González-Barahona, and J.-C. Deprez. Assessing floss communities: An experience report from the qualoss project. In *OSS*, page 364, 2009.
7. G. Madey, V. Freeh, and R. Tynan. Modeling the Free/Open Source software community: A quantitative investigation. In S. Koch, editor, *Free/Open Source Software Development*, pages 203–221. Idea Group Publishing, Hershey, Pennsylvania, USA, 2004.
8. 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*, 11(3):309–346, 2002.

[6] `http://melquiades.flossmetrics.org`
[7] `http://ossmole.sourceforge.net/`
[8] `http://www.ohloh.net/`