FLOSS Communities: Analysing Evolvability and Trustworthiness 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. This paper presents work done in the QualOSS project, which aims at building a methodology and tools to evaluate the communities around FLOSS products (among other quality attributes). Following the Goal-Question-Metric approach, QualOSS describes goals, the associated questions and then metrics whose measurements helps answer those questions. 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, quality models, product quality, data mining, libre software communities.

1 Introduction

QualOSS [1] (Quality of Open Source Software) is a research project focused on the assessment of the quality of FLOSS (free, libre, open source software)

[1] 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/

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 FLOSS software component [12]. 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. One of the key final goals of the project is to allow for the comparison of FLOSS products in a objective, semi-automated, simple and fast way. Robustness and evolvability of FLOSS endeavors are key aspects for software companies that increasingly integrate FLOSS components in their solutions and systems to produce new, reliable software application quickly.

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 FLOSS endeavor (or 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.

Most research define maintainability as a characteristic of a product. This is acceptable for certain quality characteristic such as reliability and efficiency which are real intrinsic quality characteristic of a product. Although a same approach can apply to maintainability, it often is quite restrictive. Maintainability is a characteristic that is associated to a product and the set of people involved in its implementation. This situation is further emphasized in the FLOSS world. Indeed software implementation is such a human intensive activity that the community of contributors is probably a factor as influential, if not even more influential, than the product went assessing maintainability in the FLOSS world.

The remainder of this paper is as follows: related research and state of the art in FLOSS assessment is presented. Then, the Goal - Question - Metric approach [4] is introduced. This section also deals with the definition of thresholds based on the analysis of hundreds of projects. The fourth section handles with the tools that have been used to obtain the data and presents some results from the dataset selected as case studies in QualOSS. Finally, taking into account the results section, several conclusions are drawn.

2 Related Research

According to the environment for which they have been designed we could classify quality models in two branches. Historically, most of the models were industry-oriented, focused on the development practices found in software development firms, and usually based in product and process metrics. However, with the rise of FLOSS development, some researchers considered that new approaches were needed for considering the different practices found in FLOSS projects. The QualOSS project can be framed in this second branch.

Regarding industry-oriented quality models, several well-known models exist, most notably those by McCall [22] and Boehm [3]. More recently, the quality model of the Deutsche Gesellschaft fur Qualität [1], the NASA SATC [17], or the ISO 9126 standard [2] can be cited.

In the area of FLOSS, several models have been proposed as well, such as OpenBRR[2] or QSoS[3]. They consider metrics in several realms relevant to FLOSS development and maintenance, ranging from product to process or community metrics. Their main aim is to produce a final mark (or set of marks) which shows their readiness for industrial use. Their main drawback is that, in their current state, they are not supported by automatic processes, which means that putting them to work is a tedious and time-intensive task. 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 [11]. 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 [8]. On the other hand, QSoS provides four metrics related to activity over the source code [25]

Focusing on the academic side, recent research has been carried out on this issue in the case of FLOSS endeavors. Mockus *et al.* identified that in FLOSS projects a small set of people carry on the majority of the software development (they are named as the core group [24]). Some other authors studied the composition of the core group in several FLOSS projects. They found that just few projects show a stable core group, what means that there exist a turnover of developers [26], this is unavoidable and there is a knowledge gap left by the old core group [19]. Thus, in each regeneration of developers, there are a small set of developers who lead the contributions, but for a limited amount of time [23].

QualOSS is aimed to fill this gap [18], [28], 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. It will be further explained in the next section. 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 GQM methodology, from which relevant goals, questions, and finally metrics and indicators were derived.

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

The computation of metrics measurements and aggregation for answering questions of the QualOSS quality model was partially automated with several tools.

3.1 Interviews and GQM

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 [13]. 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 pre- production state and with no stable releases. They highlight 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. The licensing model of the product is important for them, too [6].

With the information obtained from these interviews and their analysis, the ISO 9126 standard was chosen as a starting point, merging in it the criteria that we found relevant for FLOSS, producing the initial QualOSS quality model [7].

The approach for translating the inputs provided by the interviews into parts of the model was the Goal–Question–Metric methodology. Based on the stated goals, questions had to be formulated. Regarding community assessment, we had two main questions. First was “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, that is, FLOSS endeavor.

In addition, we are interested in a long term vision of the project. In fact, it is more interesting to know about the evolution trend of a FLOSS endeavor rather than to know about the current state of an FLOSS endeavor.

From this point of view, and following the GQM approach, we have defined several questions to be answered. Since this is a paper and there is limited space, just some questions will be written down. More information and specific questions can be found at [10]

Community Size and Regeneration Adequacy:

- Has the evolution of new core contributing members remained stable or grown over the history of the FLOSS endeavor? (a core member is one with commit right who perform commits frequently for instance, more than once every three month period)
- Has the evolution of core members who stopped contributing for a significant period been compensated by the joining of new core members around the same time frame?

Community Interactivity and Workload Adequacy:

– Is the number of code commits adequate (to show a lively committer community)?
– Are they sub-groups in the community? If so, are they disconnected or are active community members serving as bridges between these sub groups

3.2 Metrics

To answer the set of questions and describe the corresponding community aspects of FLOSS endeavors, the following metrics have been defined[4]:

– **sra7** average number of months where each committer committed,
– **iwa4** proportion of files maintained by a single committer,
– **iwa5** ratio of the number of lines of source code by the number of committers active in the last year,
– **iwa7** proportion of code files committed to in the last year.

Metrics describing the evolution of projects are also created. Dividing the life of a project into intervals, we measure various low level metrics for each time interval. For each project, we then capture the evolution of each low-level metric by computing the slope of the least square line fitting the resulting sequence. The following slope metrics are defined with the corresponding low-level metrics.

– **sra2** number of new code committers in the interval,
– **sra3** number of new non code committers in the interval,
– **sra9** number of active code committers in the interval,
– **iwa1** ratio of the number of commits by the number of committers in the interval,
– **iwa2** ratio of the number of code commits by the number of code committers in the interval.

Additionaly, as the notion of *core committers* was introduced, for each project, we then also compute the slope of the sequence of following low-level metrics:

– **sra4** number of new core committers in the year
– **sra5** number of core committers that quit in the year
– **sra6** difference between sra4 and sra5.

3.3 Indicators

For each metric m of a project, let us define a high-level risk indicator $r(m)$. This indicator should measure one aspect of the risk taken by a company engaging in a full FLOSS collaboration with the project. The QualOSS project has defined

[4] Each metric's id is created by the quality attribute name and its position: Community **S**ize and **R**egeneration **A**dequacy (sraX) and Community **I**nteractivity and **W**orkload **A**dequacy (iwaX)

the following four color-coded risk levels, in order of decreasing risk: black, red, yellow and green.

Defining meaningful indicators is not a trivial problem. Before we describe our methodology, let us stress 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. Thus, as an example, if M is a random variable with invertible cumulative distribution function (CDF) $F(m) = P(M \leq m)$, the *pth quantile* is defined by $F^{-1}(p)$. Hence, if m is a metric with invertible CDF, an increasing indicator $r(m)$ is defined by

- black if $m \in \left]F^{-1}(0.75), +\infty\right[$,
- red if $m \in \left]F^{-1}(0.5), F^{-1}(0.75)\right]$,
- yellow if $m \in \left]F^{-1}(0.25), F^{-1}(0.5)\right]$,
- green if $m \in \left]-\infty, F^{-1}(0.25)\right]$.

Following this rational a decreasing indicator is defined in the same way as an increasing indicator.

To compute our metrics and estimate our indicators, we use data collected by the FLOSSMetrics [16][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 open-source projects. This precludes very small projects for which, from a business perspective, a fork should be more appropriate. Hence, we choose to filter the FLOSSMetrics data projects with less than four committers during their whole life, as suggested in [15].

This particular choice of filter appears to increase the quality of our indicators. The effect of filtering on the number of projects with sufficient data in FLOSSMetrics to measure each metric is given in Table 1.

As stated before, the indicators presented above measure relative risks. This is acceptable for metrics, such as sra7, iwa4, iwa5 and iwa7, that are easy to interpret. For instance, it is intuitively plausible to say that the risk level associated to sra7 is green if the average longevity of a committer is more than a year. For the slope metrics, the situation is less satisfying. The sign of a slope is easily interpretable. However, its magnitude is less so and it is more difficult to justify intuitively the choice of indicators.

3.4 Beyond slope metrics

Let us analyze slope metrics to gain some insight. Slope metrics are based on low-level metrics that are interpretable. Using least square linear regression, we model a sequence of low-level metric values by a line $y(t) = w_0 + w_1 t$ and use

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

Table 1: Number of projects with sufficient data in FLOSSMetrics to measure some of the metrics before and after filtering, as an example

Metric	Unfiltered data	Filtered data
sra7	1422	735
iwa4	1422	735
$\mathrm{iwa1}_{90}$	1152	732
$\mathrm{iwa1}_{180}$	1095	727
$\mathrm{iwa2}_{90}$	500	289
$\mathrm{iwa2}_{180}$	470	285
sra4	1119	729
sra5	1119	729

its slope w_1 as our metric. The slope may be written as the difference between the model value at two consecutive time intervals

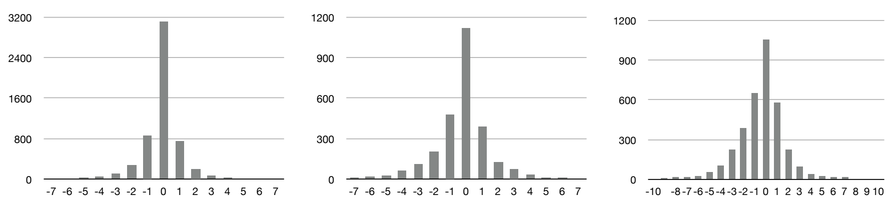
$$y(t+1) - y(t) = w_1.$$

This suggests to formulate our risk assessment problem directly as a prediction problem. Given project and a low-level metric $m(t)$, we propose to predict the difference between the value of the metric after a certain time interval and the current value. Any algorithm used to predict that difference would then define a particular high-level predictive metric.

A risk indicator is then naturally defined in terms of differences between consecutive low-level metric values. Additionally, it does not embed implicit assumptions about the particular model used for the prediction. Indeed, as demonstrated above, using the slope amounts to using a linear model to predict the low-level metric difference. This separation opens up future research directions: advanced models may be used to predict the differences more accurately, while the indicator definition stays the same.

Using the FLOSSMetrics data, let us propose indicators replacing the former slope indicators. For each low-level metric, we pooled the differences between consecutive metric values over all projects with sufficient data in FLOSSMetrics and with more than four committers. The (differences for the) low-level metrics sra2, sra3, sra9, sra4, sra5, and sra6 are integer-valued. Hence, their CDF is not invertible and we may not use quantiles to define a partition of the differences. Histograms of their differences are shown in Figure 1.

When the interval length is the same, the histograms of sra2, sra3, and sra9 are similar. Keeping the definition of those metrics in mind, we propose to use the same indicator intervals when the interval length is the same. Similarly, we propose to use the same indicator intervals for sra4, sra5, and sra6. For iwa1 and iwa2, we estimate quantiles to search for a uniform risk distribution. These new indicators defined in terms of differences are given in Table 2. Although subjective, we believe that our choice of indicators for sra2, sra3, sra9, sra4, sra5, and sra6 is intuitively reasonable. In particular, it does not lead to alarming



(a) 90 days (b) 180 days (c) sra6

Fig. 1: Histograms of the sra2 differences and sra6 differences

observed risk distributions. On the other hand, we notice that the risk distribution for the 90 days version of iwa1 and iwa2 appears to be non-uniform. Taking a close look at the data, it appears that the metric CDF is non invertible. In particular, there are 841 samples with a difference of 0 for iwa1 and 574 samples with a difference of 0 for iwa2. In both cases, to obtain a risk distribution closer to uniform, we simply suggest to move the value 0 from the red risk level to the yellow risk level. The redefined indicators are given in the rows denoted by iwa1'$_{90}$ and iwa2'$_{90}$ in Table 2.

	Indicator			
Metric	Black	Red	Yellow	Green
sra2$_{90}$	$]-\infty, -2]$	-1	0	$[1, +\infty]$
sra2$_{180}$	$]-\infty, -3]$	$[-2, -1]$	$[0, 1]$	$[2, +\infty]$
sra3$_{90}$	$]-\infty, -2]$	-1	0	$[1, +\infty]$
sra3$_{180}$	$]-\infty, -3]$	$[-2, -1]$	$[0, 1]$	$[2, +\infty]$
sra9$_{90}$	$]-\infty, -2]$	-1	0	$[1, +\infty]$
sra9$_{180}$	$]-\infty, -3]$	$[-2, -1]$	$[0, 1]$	$[2, +\infty]$
iwa1$_{90}$	$]-\infty, -19.4]$	$]-19.4, 0]$	$]0, 14.5]$	$]14.5, +\infty]$
iwa1'$_{90}$	$]-\infty, -19.4]$	$]-19.4, 0[$	$[0, 14.5]$	$]14.5, +\infty]$
iwa1$_{180}$	$]-\infty, -39]$	$]-39, -1]$	$]-1, 23.7]$	$]23.7, +\infty]$
iwa2$_{90}$	$]-\infty, -20.3]$	$]-20.3, 0]$	$]0, 13.5]$	$]13.5, +\infty]$
iwa2'$_{90}$	$]-\infty, -20.3]$	$]-20.3, 0[$	$[0, 13.5]$	$]13.5, +\infty]$
iwa2$_{180}$	$]-\infty, -37.2]$	$]-37.2, -2]$	$]-2, 18.5]$	$]18.5, +\infty]$
sra4	$]-\infty, -2]$	-1	0	$[1, +\infty]$
sra5	$[1, +\infty]$	0	-1	$]-\infty, -2]$
sra6	$]-\infty, -2]$	-1	0	$[1, +\infty]$

Table 2: Indicators defined in terms of metric differences

3.5 Tools

CVSAnalY [27] analyzes source code management (SCM) repositories. Currently it can work with three kinds of repositories: CVS, Subversion and GIT. For each of them, CVSAnalY creates a database that consists of several tables, being the most important:

– **log**: log file from Subversion, CVS or GIT.
– **committers**: committers and related information.
– **files**: files and related information.

4 Threats to Validity

As it was said, during the extraction data from FLOSSMetrics database it did not provide full of data regarding other data sources except for the SCM. It means that the results provided for some of the indicators (based on more than one specific metric) are not totally statistical-based. The indicators from mailing lists and BTS were created based on experts opinion and not based on a deep analysis of a set of FLOSS projects. However, what it is presented in this paper is just based on those metrics which are retrieved specifically from the source code management system.

It is also necessary to deal with the fact that projects stored in FLOSSMetrics database do not represent the whole population of FLOSS projects. Most of the projects stored are from Apache, GNOME, KDE and SourceForge projects, which provide a huge set of big, medium and small projects, but they, again, do not represent the FLOSS world by themselves.

Also, the statistical approach to calculate differences or tendencies in a FLOSS project may not be the best approach, however, the new definition of indicators solve this problem basing these data in just the tendency and not making it dependable from the statistical approach.

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 Results for Illustration

As an example, table 3 shows the results for a set of projects directly taken from those which have been used to create indicators. We have focused the results on the periods of 90 and 180 days since these are the longest periods of our study. The main reason for this is that for investment purposes is more interesting to know about how a community will behave during next months, than just during next days. However, the longer the period is, worse accuracy in the results is provided.

As it was said, the indicators are thought as a way to explain the current tendency of the community in terms of robustness and evolvability. So, from an industrial perspective it is more reliable the Evince project than the Nautilus, since it has better indicators ("more greens"). It denotes that there is a problem in Nautilus community and there are some risks that may be studied deeper. Going a step ahead, if the metrics are checked, during last years, the Nautilus projects has decreased its number of committers and also its number of contributions, what means that the maintenance done before is slightly decreasing during last months. It is also important to notice that the core committers (those which do most of the work in a given community) have a greater behavior from that industrial perspective than the Nautilus'. Thus, we may conclude looking at table 3 that there are more intrinsic risks on the Nautilus community than in the Evince community. It may be useful, for instance, to decide if the Nautilus community needs more effort to go on with their maintenance activity or if your company prefer to invest money on a growing community.

Metric	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
sra2$_{90}$	-0.1008 B	0.0346 G	-0.0701 R	-0.0463 R
sra2$_{180}$	-0.3958 B	0.1015 G	-0.3072 R	-0.1839 R
sra3$_{90}$	-0.079 R	0.2457 G	0.0214 G	-0.0474 Y
sra3$_{180}$	-0.2987 R	0.8158 G	-0.0045 G	-0.1891 Y
sra9$_{90}$	0.0257 Y	0.1379 G	-0.0735 R	-0.276 B
sra9$_{180}$	-0.0203 Y	0.3609 G	-0.2445 B	-0.636 R
iwa1$_{90}$	-0.23327969 Y	-0.0312513 Y	-0.20157016 Y	-0.62113349 R
iwa1$_{180}$	-0.61774331 Y	-0.24498722 Y	-0.73020034 Y	-2.19266004 R
iwa2$_{90}$	-1.03448107 R	0.31998312 Y	-1.60891953 R	-0.76343814 Y
iwa2$_{180}$	-2.47361739 Y	-0.15635451 Y	-5.50437436 R	-2.71498719 Y
sra4	0.5 G	1.5636 G	0.1091 G	-0.3582 R
sra5	1.2364 B	1.4848 B	1.5091 B	0.0132 Y
sra6	-0.7364 R	0.0788 G	-1.4 B	-0.3714 R

Table 3: Illustration in some projects using slope approach

On the other hand, using the linear model's approach, Table 4 shows the new set of results. For this approach, there is a new notation based on Δ_X: Δ_X sra2$_Y$ where Δ_X means that we predict the metric difference over the next X days and sra2$_Y$ means that we use a linear model estimated with an interval length of Y days

For example, if the model for iwa1 built with an interval of 30 days is iwa1 = 0.2 - 0.1 t. Then we predict a difference over 30 days of -0.1, over 90 days of

-0.3=-0.1*3 and over 180 days of -0.6=-0.1*6. If the metric we want to predict is discrete, then we round the continuous difference prediction to the closest integer.

This approach shows different results from the previous one. In this case, all the communities show a similar behavior in terms of tendency. If we check Table 3 the differences in the slope for all the projects are not really huge. Thus, it makes sense, since there is an approximation to the closest integer as aforementioned. We believe that this new approach polishes the first one and results are more accurate to the real world. 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
Δ_{90} sra2_{90}	0 Y	0 Y	0 Y	0 Y
Δ_{180} sra2_{180}	0 Y	0 Y	0 Y	0 Y
Δ_{90} sra3_{90}	0 Y	0 Y	0 Y	0 Y
Δ_{180} sra3_{180}	0 Y	1 Y	0 Y	0 Y
Δ_{90} sra9_{90}	0 Y	0 Y	0 Y	0 Y
Δ_{180} sra9_{180}	0 Y	0 Y	0 Y	-1 R
Δ_{90} iwa1_{90}	-0.23327969 R	-0.0312513 R	-0.20157016 R	-0.62113349 R
Δ_{180} iwa1_{180}	-0.61774331 Y	-0.24498722 Y	-0.73020034 Y	-2.19266004 R
Δ_{90} iwa2_{90}	-1.03448107 R	0.31998312 Y	-1.60891953 R	-0.76343814 R
Δ_{180} iwa2_{180}	-2.47361739 R	-0.15635451 Y	-5.50437436 R	-2.71498719 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 4: Illustration in some projects using a linear model

6 Conclusions

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. Taking into account the latter, we may not be interested in adding more effort in a given project due to the fact that main developers are leaving the project and this is a tendency checked during last months or years. On the other hand, 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. It is also important to check the status of potential activity per committers. Sometimes it is necessary, for maintenance purposes to check how many files are being handled by committers in order to check if they are overloaded.

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.

7 Further Work

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 projects is currently adding data from BTS 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 [24, 14, 21]. Even, more complex analysis regarding the network structure [20] of communities could also be the base for providing additional objective metric related to the community.

Acknowledgment

We thank the anonymous reviewers for their comments. 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

References

1. Deutsche Gesellschaft fuer Qualitaet, Software-Qualitaetssicherung. Technical report, VDE-Verlag Berlin, 1986.

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

2. *ISO/IEC 9126 International Standard, Software engineering-Product quality, Part 1: Quality model.* 2001.
3. E. al. Barry W. Boehm. *Characteristics of Software Quality (TRW series of software technology).* Elsevier.
4. V. R. Basili. Software modeling and measurement: the goal/question/metric paradigm. Technical report, College Park, MD, USA, 1992.
5. B. W. Boehm. Software risk management: Principles and practices. *IEEE Softw.*, 8(1):32–41, 1991.
6. M. Ciolkowski, M. Soto, and J.-C. Deprez. Measurement requirements specifications. specification of goals for the qualoss quality model. Technical report, QualOSS Consortium, 2007.
7. M. Ciolkowski, M. Soto, and J.-C. Deprez. Metrics system and prototype qualoss models. Technical report, QualOSS Consortium, 2007.
8. O. Consortium. Business readiness rating for open source. Technical report, Carnegie Mellon West Center for Open Source Investigation, CodeZoo, SpikeSource and Intel, USA, 2005.
9. T. De Marco and T. Lister. *Peopleware : Productive Projects and Teams, 2nd Ed.* Dorset House Publishing Company, Incorporated, 1999.
10. J.-C. Deprez. Reference f/oss project report. Technical report, QualOSS Consortium, 2008.
11. 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.
12. 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.
13. J.-C. Deprez, J. Ruiz, and I. Herraiz. Evaluation report on existing tools and existing f/oss repositories. Technical report, QualOSS Consortium, 2007.
14. 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.
15. I. Herraiz. *A statistical examination of the evolution and properties of libre software.* PhD thesis, Universidad Rey Juan Carlos, 2008. http://purl.org/net/who/iht/phd.
16. I. Herraiz, D. Izquierdo-Cortazar, and F. Rivas-Hernández. Flossmetrics: Free/libre/open source software metrics. In *CSMR*, pages 281–284, 2009.
17. L. Hyatt and L. Rosenberg. A software quality model and metrics for risk assessment. In *ESA'96 Product Assurance Symposium and Software Product ASS.*
18. 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.
19. D. Izquierdo-Cortazar, G. Robles, F. Ortega, and J. M. Gonzalez-Barahona. Using software archaeology to measure knowledge loss in software projects due to developer turnover. *Hawaii International Conference on System Sciences*, 0:1–10, 2009.
20. L. López, G. Robles, J. M. G. Barahona, and I. Herraiz. Applying social network analysis techniques to community-driven libre software projects. *International Journal of Information Technology and Web Engineering*, 1(3):27–48, July-September 2006.
21. 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.
22. J. A. McCall, P. K. Richards, and G. F. Walters. Factors in software quality. Technical report, 1977.

23. M. Michlmayr, G. Robles, and J. M. Gonzalez-Barahona. Volunteers in large libre software projects: A quantitative analysis over time. In S. K. Sowe, I. G. Stamelos, and I. Samoladas, editors, *Emerging Free and Open Source Software Practices*, pages 1–24. Idea Group Publishing, Hershey, Pennsylvania, USA, 2007.
24. 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.
25. A. Origin. Method for qualification and selection of open source software. Technical report, Atos Origin, France, 2006.
26. G. Robles. Contributor turnover in libre software projects. In *Proceedings of the Second International Conference on Open Source Systems*, 2006.
27. G. Robles, S. Koch, and J. M. González-Barahona. Remote analysis and measurement of libre software systems by means of the CVSAnalY tool. In *Proc 2nd Workshop on Remote Analysis and Measurement of Software Systems*, pages 51–56, Edinburg, UK, 2004.
28. M. Soto, D. Izquierdo-Cortazar, and M. Ciolkowski. Measuring the performance of open source development communities: The qualoss approach. In *DASMA Metrik Kongress*, 2009.