Two Evolution Indicators for FOSS Projects

Etiel Petrinja, Giancarlo Succi

Two evolution indicators for FOSS projects

Etiel Petrinja[1] and Giancarlo Succi[1]

Free University of Bozen/Bolzano

Abstract

In this paper we introduce two project evolution indicators. One is showing an increase of downloads of the project and therefore a growing interest of users in the results of the project. The second indicator is predicting the future evolution of the project with the submission of new revisions to the concurrent versioning system. Both indicators can provide evidence of the sustainability of a software project. We used the General Linear Model method to statistically formulate the two linear equations that can be used to predict the two indicators. The predicting equations were build by using two stratified data samples one of 760 projects and the second of 880 projects extracted from the SourceForge repository. The six metrics included into the final version of the two models were extracted from a set of thirty project and product metrics as: the number of downloads, the number of developers, etc. We have validated the discriminant and the concurrent validity of the two models by using different statistical tests as the goodness-of-fit and we have used the two models to predict the indicators on two hold-out validation samples. The model predicting the increment of downloads was correct in 75 percent of the cases, the model predicting the submission of new revisions was correct in 93 percent of the cases.

1 Introduction

Software projects evolve according to different evolution processes. The Free Source Software [9] projects or the Open Source Software projects (FOSS) evolution differs in several aspects from closed source software (CSS) projects [11]. The success of closed source software projects is usually correlated with the number of copies of the software product soled [27]. FOSS projects are often considered successful if there is a large number of users of the FOSS product [18]. As proposed by DeLone and McLean in their Model of Information systems Success paper [28] the number of FOSS users can represent the 'Users satisfaction' factor that is included in their model. The number of users depends on the quality of the software product and on the quality of the development process that is followed inside the FOSS community [18], [6].

The number of FOSS projects has been growing rapidly in the last decade [29]. The aim of the presented research was to identify characteristics of successful FOSS projects focusing on the number of downloads of FOSS products and the vitality of the development process. Based on those characteristics we

defined two indicators of the future evolution of the FOSS project. The success of a software product is a common dependent variable of the research of software projects, and due to its dependency on several aspects, it should be modelled as a multidimensional factor. The number of downloads should be considered as a metric indicating the interest of users into a FOSS product, not necessarily its real usage. Not all downloaded projects are used and there is also a large number of users of FOSS products that have not downloaded them but obtained them as part of a software bundle; for example in one of the Linux distributions. Taking in consideration these two deviations, the number of downloads of FOSS projects is still a factor that is well correlated with the number of users of a FOSS product. The research presented in this paper was based on a empirical study of several thousands FOSS projects stored in the SourceForge repository. The characteristics of FOSS projects were identified by defining and validating two indicators of FOSS project evolution based on a set of thirty measurements. Two models were studied: one is focused on the interest of users in the FOSS product, the second is explaining the potential future survival of the FOSS project by inserting new revisions (working increments of the project) into the project's versioning system. Another important characteristic of FOSS projects is the sustainability. It depends on a large number of product and process aspects. The two indicators, presented in this paper, can provide evidence of the sustainability of the FOSS project. An growing number of users and a high probability of new revisions being published is correlated with the sustainability of the project. The data collected about the projects stored in SourceForge were used for building and validating the two models following a statistical approach.

Section two provides background information reviewed for the study. Section three contains information about the measures collected, the sources of information used, the building of the two evolution indicators, and their validation. In section four we discuss the results and consider the limitations of this study. Finally, in section five we conclude the paper.

2 Background

The quality of a FOSS product is an important factor considered when adopting it or planing to start contributing to the project. The open availability of project data has supported a large set of FOSS studies. Many of those studies were focused on software measurement and especially on the measurement of its quality. Scacchi [12], McConnell [26], Alshayeb and Li [10] published observations of the results of studies comparing Closed source software projects and FOSS projects showing the differences and similarities between the two development approaches.

The Lines of Code (LOC) and their variation through time was a metric often considered and analysed [22], [11]. Crowston and Scozzi [13] analysed the FOSS development process and proposed transitions between different phases of

the FOSS development process. Several studies were conducted by statistically analysing a large sample of projects. Capiluppi et al. [15] conducted a horizontal study of 404 FOSS projects focusing on the change of the size of alive projects. They observed that only a small percentage of projects is in a growing phase. Similar studies analysing large samples of FOSS projects were done by Koch [16] that identified a relationship between the size of the project, the number of participants, and the distribution of work in the development team. Robles-Martinez et al. [17] studied the MONO project by taking in consideration the lines of code, the commits, and the authorship of contributions.

Projects available on SourceForge were used for FOSS studies published by for example Grewal et al. [20], Koch [16], or Capiluppi et al. [15]. Raja and Tretter [25] have used the data of 290 projects available on SourceForge to build and validate a model of FOSS projects survivability. Their study allowed them to propose a stable model to define the survivability of a FOSS project. The model is based on three factors: the organization of the project, the resilience of the community, and its vigour. We have adopted a similar approach to build our models, however along the three factors we have included in the study several other FOSS software and process metrics.

Various publicly available data sources were used for studies of FOSS. The source code stored in versioning systems is one of them. Code level studies done by Godfrey and Tu [22] and Robles et al. [23] are often based on concurrent versioning system tools as CVS, Subversion, and recently GIT or Mercurial. Mishra et al. [24] proposed a quality model that analyses factors contributing to code quality, such as the number of developers, or the mix of talents involved in a FOSS community. Other studies include data from the issue management system as the number of issue contributors, the time necessary to solve an issue [25], etc. Mailing lists [23] and the information available on web pages are also an important source of information for studies on FOSS projects Koch [16].

Crowston et al. [18] proposed a framework for measuring the success of FOSS projects. The exact number and type of users of FOSS products is not well-known as it is in commercial projects, where customers are usually well profiled. Crowston et al.[19] proposed to insert an automatic feedback collection mechanism directly inside the software product. This functionality has been implemented in the last years in projects as Firefox, Ubuntu, Libre Office, and others. Grewal et al. [20] identified user and technical criteria for inferring the success of FOSS projects. In part similarly to our approach Polancic et al. [21] proposed a framework for evaluating OSS projects based on simple metrics.

The FOSS development process was studied by Taibi et al. [5] and Petrinja, et al. [6]. They proposed evaluation models as the MOSST and the OMM models based on FOSS product and process metrics. Deprez and Simons [8], and Petrinja et al. [7] have compared the models used to assess the FOSS development process and identified the critical aspects of the analysed models. Some of the issues identified were the subjectivity of the assessment process, and a difficult interpretation of some metrics. In the study presented in this paper we use a set of metrics to propose evaluation models that are based

on a statistically significant sample of FOSS projects and are not biased by a subjective interpretation of factors.

3 Analysing project evolution

Based on a sample of project data extracted from concurrent versioning systems and web pages available on the SourceForge repository, we sudied which characteristics influence the number of downloads of a FOSS product, and which factors can be used to predict the evolution of the FOSS project. Based on identified characteristics we proposed two evolution indicators. We inferred the evolution of projects by counting the number of new revisions inserted into the versioning system. The number of downloads is indicating the user's interest in the project. New revisions indicate the further evolution of the FOSS project and therefore the interest in the project by the FOSS community. The two aspects shows the expansion or restriction of the development process. We considered the increase of the two aspects as an indication of improved quality of the FOSS project. We defined a dichotomous value for both characteristics and calculate them for each FOSS project included in the study. The Equation 1 shows the definition of the download increment factor and the Equation 2 shows the revisions indicator. In both cases k represents the time period. We decided to choose one year as the studied time period, in particular we used the data available for year 2011. For building the model and calculating the metrics we used all the data available for the considered projects from when they have started. Some projects included in the study exist already for more than ten years.

$$DownloadsIncrement(D_i) = \begin{cases} 1 & \text{if } \sum_k (Downloads_k - Downloads_{k-1}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$Revisions(R_i) = \begin{cases} 1 \; if Revisions_k > 0 \\ 0 \; otherwise \end{cases} \tag{2}$$

The generalized linear model (GLM) is in statistics the generalization of the linear regression approach. It allows linking the variance of each factor included in the model (independent variables) to be linearly connected with the predicted value (the dependent variable). The dependent variable can be calculated using the Equation 3. The predicted value is the sum of the independent factors multiplied by a weighting factor β summed with an intercept constant.

$$PredictedValue = \sum_i \beta_i * IndependentFactor_i + Intercept \tag{3}$$

3.1 Data sources

The SourceForge project repository contains currently nearly 360.000 projects and exists already for more than a decade. In spite of the high number of projects only a small percentage of those projects is still alive and evolving. The large majority of them is represented only by a name, and a brief description of an idea to be implemented inside the project. Half of the projects in SourceForge adopted the Subversion versioning system and the other half is using the CVS. The GIT, the Mercurial, and the Bazaar versioning systems are not yet largely used inside SourceForge. Out of the 150.820 projects using Subversion only 66.674 have at least one revision inserted into the versioning system. In several cases the projects that are inserted into SourceForge have also an external database and versioning system. The projects are not always promptly synchronised between the two repositories. A retrieval of external project data and comparison with the data available in SourceForge could provide an interesting indications of the life cycle of those FOSS projects. We limited our study just to the data stored in different data sources all composing the SourceForge system.

We have decided to considered just the projects stored in the Subversion system. We have collected different data as the number of revisions and the date when they were inserted, for each of the 66.674 projects with at least one revision. Another important source of information about the projects were the SourceForge web pages presentations available for each project. We have spidered the pages collecting metrics as: the name of the project, the staring date, the status of the project, the issues reported and solved, the developers involved in the process, the number of downloads, the time distribution of the downloads, and others. Joining all the data collected we have limited the number of projects useful for our study. The number of projects with all the characteristics necessary for our study stored in SourceForge was 5.905. We have performed several analyses on these data and we have additionally limited their number for some studies. For the two studies reported in this paper we have considered only projects that exist for a period longer than 1 year when calculating the download increase factor and for a period longer than 3 years for building the model related to the revisions and survival of the FOSS project. The factors included into the study are simple project metrics that do not need additional explanations. Three composed factors: vigour, resilience, and organization have been calculated following the equations proposed by Raja and Tretter [25].

3.2 Building the two models

We aimed to predict if the number of downloads in the considered period will be higher from the preceding period. We predicted this characteristic (dependent variable) from a set of factors characterising FOSS projects (independent variables). For building the model we adopted the General Linear Model (GLM).

We used the R statistical evaluation tool to automate the GLM calculation. After restricting the projects to the one that exist for more than one or three years we have obtained a sample (with 760 projects) as inputs for building the downloading evolution model. Based on the number of downloads extracted from the SourceForge repository for the years 2010 and 2011 we calculated the downloading delta factor assigning a 1 if the number of downloads was higher in 2011 than in 2010 and 0 if the number was lower. For improving the correctness of the accuracy, and the precision of the proposed model it is important that the number of projects that have increased the number of downloads (having the download increase factor 1) and those that did not (having the download factor 0) should be equal. In our initial sample of projects the number of projects that did not increase the number of downloads was larger than the number of projects increasing the number of downloads. We had to limit the number of the first type of projects. We randomly selected projects from the first group to reach the number of projects in the second group. We have prepared both a modelling and a hold-out validation sample following the stratified sampling approach. We first selected all projects that exist for more than one year and divided them into two strata based on the increment of the number of downloads. We used two-thirds of the sample for building the model and one-third for validating the model.

We have considered 30 different factors for building the download increment model and 12 factors for building the model of the liveliness of the project by predicting the future revisions of FOSS projects. Due to space constraints we are unable to present in details all the metrics considered. However just few of the factors proved to be significant for building the two models. Four factors were dichotomous (with values 0 or 1) the other represented continuous metrics. In the continuation of this section we present basic statistics of the factors included in the final versions of the models developed.

The two dichotomous factors that were included into the download increment model were ActivityRev2011 and ActivityDowDelta2010. The first indicates either there were new revisions for the project in year 2010 and the second indicates if there was an increment of the number of downloads in the year 2010 in comparison with the number of downloads in the year 2009. The two continuous factors included into the model were: the number of new revisions inserted into the versioning system during the year 2010 (Rev2010) and the number of new opened issues in the year 2011 (OpenedIssues2011). The majority of characteristics of projects stored in SourceForge is strongly skewed to the right.

The ‘step‘ function available in the R statistical tool was used to find the optimal combination of factors that should be included into the model. The ‘step‘ function proceeds stepwise to identify the GLM model that has the lowest Akaike Information Criterion (AIC) number. The AIC factor indicates how well the data values predicted with the help of the parametric model fits the measured data. Better the model, better the predicted data, therefore smaller the difference between the predicted and the measured data. The optimal linear model for predicting downloads increment contained 9 different factors (activ-

ityRev2010, ActivityRev2011, activityRev2011delta, closedissues2010, averageclosuretime2010, openedissues2011, closedissues2011, averageclosuretime2011, and activityDow2010Delta). We limited the number of factors to four by loosing less than one percent of the prediction power of the model. We limited the number of metrics that have to be collected to improve the usability of the model. The statistical data characterising the factors included into the model are presented in Table 1. We see that the mean value of revisions in 2010 is 103 per project. There are some projects with zero revisions and one with 3431 revisions. The number of issues reported in the year 2011 for our sample of projects is not very large; the larger is 86 and the mean is just 2.6 issues. The percentage of projects that have more revisions in 2011 then in 2010 in our sample is 65 percent. The DownloadDelta Checker for 2011 is balanced for the correctness of the prediction as discussed previously.

We have designed the final GLM model by considering the prediction power of singular factors. We obtained a list of β factors for each of the 30 factors considered for predicting the downloading increment. The prediction power of factors were: projectLongevity (0.54), revisionsTotal (0.518), revisions2010 (0.575), revisions2011 (0.602), activityRev2010 (0.625), activityRev2011 (0.682), activityRev2011delta (0.595), vigorAverage2010 (0.589), vigorAverage2011 (0.607), openedissues (0.501), closedissues (0.501), openedissues2010 (0.587), closedissues2010 (0.595), averageclosuretime2010 (0.586), resilence2010 (0.570), openedissues2011 (0.616), closedissues2011 (0.595), averageclosuretime2011 (0.6), resilence2011 (0.597), organization (0.509), downloads (0.505), downloads2009 (0.501), downloads2010 (0.505), activityDow2010Delta (0.736), recomendedBy (0.506), createdbynumber (0.502), closedbynumber (0.504), averageclosuretime (0.516), averageclosuretimeabsolute (0.509), and numberOfContributors (0.509).

If for the download increment model we consider just one factor, we are able to predict correctly the percentage of projects shown in the brackets. We see that the increase of the number of downloads in year 2010 comparing it with the number of downloads in the year 2009 (activityDow2010Delta) factor is able to predict correctly the increase in the year 2011 in 73 percent of cases. With the combination of additional factors we tried to obtain a better prediction of the downloading model. We run the stepwise calculation of the optimal prediction model and obtained a model with four factors that are shown in Table 6. We see that the activityDow2010Delta has a strong influence in the prediction model. This is evident from the size of the β factor (1.83). The other factors have a smaller influence. Nevertheless the β factors of the Revisions 2010 factor and the OpenedIssues 2011 factor are small, they can still contribute considerably to the value of the prediction, if the number of revisions or the number of newly reported issues is large.

Before building the revision prediction indicator we have first analysed the prediction power of 12 factors singularly: projectLongevity (0.57), revisionsTotal (0.59), revisions2010 (0.74), activityRev2010 (0.71), vigorAverage2010 (0.69), openedissues2010 (0.50), averageclosuretime2010 (0.50), downloads2009

Table 1. Descriptive statistic data of the factors used for building the download incrementing indicator

	Revisions 2010	OpenedIssues 2011	Revisions Checker 2011	DownloadDelta Checker 2010
Mean	103.20	2.6	0.65	0.51
Median	18.00	0.00	1.00	1.00
Std Dev	252.99	8.17	0.48	0.50
Min	0.00	0.00	0.00	0.00
Max	3431.00	86.0	1.00	1.00
N	760	760	760	760

Table 2. Descriptive statistic data of the factors used for validating the download incrementing indicator

	Revisions 2010	OpenedIssues 2011	Revisions Checker 2011	DownloadDelta Checker 2010
Mean	127.10	3.77	0.67	0.51
Median	23.00	0.00	1.00	1.00
Std Dev	276.02	20.44	0.47	0.50
Min	0.00	0.00	0.00	0.00
Max	2444.00	352.00	1.00	1.00
N	379	379	379	379

(0.50), downloads2010 (0.50), createdbynumber (0.50), closedbynumber (0.52), and averageclosuretime (0.50). In brackets is shown the percentage of correctly predicted revision activity. We see that the number of revisions in the previous year and the checker of revision activity in the year 2010 have a high prediction power. Applying the step-by-step calculation of the optimal set of factors for best constructing the GLM revision model we identified four factors: the projectLongevity, the revisions2010, the activityRev2010, and the vigorAverage2010. The activityRev2010 is a dichotomous factor (values are 0 if there is no revision in a specific year or 1 if there is at least one revision), the other three factors are continuous metrics. We have subsequently trimmed the number of factors for simplifying the model. With the trimming the model has loosed less than one percent of its prediction power. Table 3 shows the descriptive statistics for the two factors used to build the model and Table 4 the descriptive statistics of the sample used for validating the revision model.

Table 5 shows the mutual correlations between the four factors used to define the download increment model. We calculated the Pearson correlation coefficient. We can consider the factors weakly correlated if the correlation is smaller than 0.4. As we can see in Table 5 all mutual correlations fulfil this requirement. All the tests were highly significant with the p value that was

Table 3. Descriptive statistic data of the factors used for building the revisions indicator

	Revisions 2010	Vigor average 2010
Mean	101.50	107.90
Median	6.50	10.95
Std Dev	294.37	296.09
Min	0.00	0.00
Max	3431.00	3431.00
N	880	880

Table 4. Descriptive statistic data of the factors used for validating the revisions indicator

	Revisions 2010	Vigor average 2010
Mean	109.2	120.10
Median	5.0	8.297
Std Dev	368.63	386.26
Min	0.0	0.0
Max	4337.0	4337.0
N	440	440

smaller than 0.00001 in all cases. We can consider the factors independent and therefore it is not superfluous including them all into the same prediction model.

Table 5. Correlation table of the factors used for constructing the download incrementing model

	Revisions 2010	OpenedIssues 2011	Revisions Checker 2011	DownloadDelta Checker 2010
Revisions 2010	-	0.364	0.276	0.157
OpenedIssues 2011	0.364	-	0.169	0.150
Revisions Checker 2011	0.276	0.169	-	0.305
DownloadDelta Checker 2010	0.157	0.150	0.305	-

Table 6 shows the β weights for the factors included into the download incrementing model. The biggest the β weight the largest is the influence of the

related independent variable to the dependent predicted value. We see that the download delta checker from the previous year (2010) and the revisions checker from year 2010 have a strong influence on the predicted variable.

Table 6. General linear model coefficients for the download incrementing model

	Estimate	Std. Error	z value	Pr(> |z|)
(Intercept)	-1.9040894	0.1774413	-10.731	<2e-16
Revisions 2010	-0.0009697	0.0003615	-2.683	0.00730
Revisions Checker 2011	1.4808722	0.1935980	7.649	2.02e-14
OpenedIssues 2011	0.0351792	0.0135808	2.590	0.00959
DownloadDelta Checker 2010	1.8329578	0.1749270	10.478	<2e-16

Using Equation 3 and the calculated weighting factors we can now write explicitly the linear equation of the model for predicting the increment of the downloads of a FOSS project as follows:

$$\begin{aligned} Increase of downloads = & -1.904 - \\ & 0.001 * (Revisions2010) + \\ & 1.481 * (RevisionsChecker2011) + \\ & 0.035 * (OpenedIssues2011) + \\ & 1.833 * (DownloadDeltaChecker2010) \end{aligned}$$

Similarly as for the download model we have calculated the weighting factors for the revisions model. The Equation 2 can be used to predict the probability that the project will have new revisions in the following time period (in the year 2011 in our case).

$$\begin{aligned} Increase of Revisions = & 1.201 + \\ & 451.49 * (VigorAverage2010) - \\ & 452.252 * (Revisions2010) \end{aligned}$$

We can test the statistic significance of the weighting factors β by calculating the Wald statistics and checking the values obtained. In Table 7 we see the β values, their standard error, the Wald statistic, the p values, and the exponential factors of the β values which show how strongly each factor contributes to the change of the predicted value. We see that the p-values for all factors are marginal therefore the factors are statistically significant.

To be able to predict the increase of the downloads or the insertion of new revisions we have to define the threshold value of the prediction Equation

Table 7. Wald statistic for the download incrementing indicator

	β	St. Er.	Wald	df	Sig.	Exp(β)
(Intercept)	-1.9040894	0.1774413	115.2	1	0.0	0.1489582
Revisions 2010	-0.0009697	0.0003615	7.2	1	0.0073	0.9990307
Revisions Checker 2011	1.4808722	0.1935980	58.5	1	0.0	4.3967788
OpenedIssues 2011	0.0351792	0.0135808	6.7	1	0.0096	1.0358053
DownloadDelta Checker 2010	1.8329578	0.1749270	109.8	1	0.0	6.2523523

3 for which the model will predict the increase. We predicted the threshold value by drawing the receiver operating characteristics (ROC) curve for the prediction model and finding the point where the successful prediction of the model was reached. This value is optimal when the sum of the probability of a correct prediction of the model (the sensitivity of the model) and the correct prediction of the missing of the required conditions (the specificity of the model) is maximized. The ROC curve helps to graphically identify the maximum of both values (sensitivity and specificity). In the case of the model for download increment prediction this value is 0.43. After computing the Equation 3 for the values of a specific FOSS project, if the value is higher than 0.43, the model predicts that the number of downloads in a specific year will be higher than in the previous year.

3.3 Validating the models

With the selection of only weakly correlated factors we guaranteed the discriminant validity of the two proposed models. The concurrent validity of the models can be tested by checking the goodness-of-fit of the two models and by comparing the measured and the predicted values. The goodness-of-fit of the model shows how well the prediction model identifies the correct values. Several tests exist that consider the difference between the observed and the predicted values. We have used three different tests.

The -2 Log Likelihood ratio is a statistical test for comparing the fit of the model to real data. For the model predicting the download increment the Log likelihood ratio is 224. The second test we have used was the Cox and Snell R^2 which gave for the download model the value 0.18. Which shows that the model has not a strong prediction power. The third test was the Nagelkerke R^2 test which gave for the download model the value 0.24. Based on the results of these tests we can not expect a high prediction power of the proposed download prediction model. We can see exactly how precisely the model can predict the values by using first the sample of FOSS projects used to build the model. Afterwards we will use the hold-out validation sample that was composed by one third of the initial sample.

Table 8 contains the measured and the predicted values about the increase of the number of downloads between the years 2010 and 2011 for 760 FOSS

projects. We can see the percentages of correct and missed predictions. The overall prediction precision is 75 percent and it is far from mere guessing (which would be the case if the percentage rate would be 50 percent) but it is still not very precise.

Table 8. Classification table for the download incrementing indicator

	Predicted increase of downloads		
Measured increase of downloads	Yes	No	%
Yes	294	85	77
No	107	274	72
%	73	76	75

The -2 Log Likelihood ratio for the revision download model is 177.56, the Cox and Snell R^2 value is 0.63, and the Nagelkerke R^2 value is 0.83. The three likelihood tests gave good results for the revision model showing that the model fits well to the measured data. Table 9 shows the results of the prediction and the measurement of the data used to create the Revision prediction model. The number of FOSS projects included in this test was 880.

Table 9. Classification table for the Revision prediction indicator

	Predicted new revisions		
Observed new revisions	Yes	No	%
Yes	378	63	86
No	1	438	99
%	99	87	93

The predictions shown in the classification table 8 are biased while we have used the same data to build the prediction model. By applying the download prediction Equation 3 on the validation sample of 379 FOSS projects we have obtained the results presented in Table 10. The percentages are comparable to the data used for model creation, they are just slightly lower.

By applying the new revisions prediction Equation 2 on the validation sample of 423 FOSS projects we have obtained the results presented in Table 11.

Table 10. Validation table for the download increasing indicator

	Predicted increase of downloads		
Measured increase of downloads	Yes	No	%
Yes	113	78	60
No	36	152	81
%	76	66	70

Table 11. Validation table for the new revisions prediction indicator

	Predicted new revisions		
Observed new revisions	Yes	No	%
Yes	166	35	83
No	6	216	97
%	97	86	90

4 Discussion

The further evolution of the FOSS project from the point of view of the user and from the point of view of the community provide two indications of the success of the project. It is important to take in consideration that FOSS users are sometimes also FOSS developers and that almost always the developers are also users of products they have contributed to develop.

The results of the validation show that the two prediction models can provide hints on the further evolution of the FOSS project. The equation for predicting an increased download number is less precise than the equation for predicting the availability of new revisions related to a FOSS project. One reason for this can be a higher uncertainty of the number of downloads in comparison with the number of new revisions. FOSS products are downloaded by individuals that are not always part of the FOSS community and it is therefore more difficult to predict precisely if their number will increase. On contrary the availability of new revisions depends on the past development and stability of the development process.

If the project is downloaded by a growing number of users it means that the community implements functionality that is needed and considered useful by a growing number of users. A larger number of users can afterwards provide new bug/issue reports, forum entries, or even code contributions. A growing number of downloads is a good indicator for the future development of a FOSS project. We were able to predict the future availability of new revisions more precisely than downloads. The proposed revisions prediction equation is precise and can identify projects that will stop evolving and the one that will have new revisions

in the coming time period. Having an indication of the future availability of new revisions can be an important factor when deciding to download and start using a FOSS product. If the project is not going to be improved further with new revisions, the new bugs/issues reported might not be addressed and the new features proposed will never be implemented.

We can compare our results with the results reported by [25]. They achieved accuracy of 92.78 percent for predicting the survivability of projects they have included into their training and testing samples. We obtained a similar precision (93 percent) for the prediction of new revisions and a lower precision (75 percent) for the prediction of an increase of downloads. Our testing and training samples were between three to four times larger than the one used by [25]. A large sample allows us to be more confident when generalizing the results of our study.

The factors included into the two equations show also which aspects of the FOSS project influence the further evolution of the project. An increased downloading of the FOSS product depends on new revisions in the analysed time period. If there are no new revisions inserted into the versioning system most probably the number of downloads will decrease. If the project was downloaded in 2010 more often than in 2009 it will be probably downloaded more often in 2011 than in 2010. The majority of this type of projects are in a growing phase and they are attracting an increasing number of new users. The temporal existence of these type of projects was statistically shorter than the average existence of analysed FOSS projects. It means that they are new and they are growing. A large number of new issues reported by the user base is triggered by a larger number of users and downloads. The submission of new revisions depends on the number of revisions in the previous time period and the average vigour in the previous time period. The vigour is obtained from the number of revisions in a specific time interval and it shows how strongly the project is evolving.

4.1 Limitations

The construct validity is focused on the dependent and the independent factors and how accurately they are able to model the hypothesis. Most of the measures analysed are simple as the number of downloads, the number of revisions, or the number of issues reported. Problematic could be the assumption that an increased number of downloads or the submission of new revisions indicates a higher quality of the FOSS project. The number of users of a FOSS project is not equal to the number of downloads, however an increased download rate leads to an increased number of users. The same is true for the number of revisions. The total number of revisions is not an absolute indicator of the quality of the project, nevertheless it is indicating an evolution trend of the project. Another factor used was the number of issues which are usually bug reports or new feature requests. We did not distinguish between the two, however this was not an issue for our study, since both contributions indicate an interest of the

users in the project. The factor organization and the factor resilience where not included into the final versions of the two models only the vigour factor is used for predicting new revisions. The validity of the three factors was demonstrated by Raja and Treter. Based on simply measured factors we have calculated four dichotomous measures about the change of downloads or revisions. These four measures do not present construct validity issues.

The internal validity is related to the link between the dependent and the independent variables. If the independent variables changes also the dependent variable should change. We have done rigorous testing of the factors included into the two models and the internal validity proved not to be an issue for this study.

The external validity is focused on the applicability of the results of the study to FOSS projects not included into the study. All projects used for the study have been extracted only from the SourceForge repository. This could prevent the applicability of the two equations on projects developed in other environments. However SourceForge is one of the largest and oldest FOSS projects repository and the quality of data available is good. The two models have been designed based on several hundreds of relevant projects and validated with two completely separate hold-out validation samples. Therefore we are confident that if the basic data necessary for the prediction is extracted diligently, the models should be valid also on FOSS projects contained in other FOSS repositories. An extended study replicated also on other FOSS repositories could anyway benefit the external validity of the proposed models.

5 Conclusions

In this paper we have presented an analyse of a large set of FOSS projects and identified two prediction equations. One for predicting the increase of product downloads and one for predicting the further development of the project with new revisions being submitted to the source code versioning system. The two indicators can provide a hint on the sustainability of the FOSS project. Based on simple project metrics users can understand if the project will evolve in the near future. The two predictions can be useful for the FOSS community that is developing the project and also for potential new users of the FOSS product. The community can benefit from the information about a potential risk of a diminishing number of downloads and can take preventive actions. A new user can decide to start using a product if there is a good chance of its further evolution proved by the probability of new revisions published in the future. The study presented in this paper is building on top of several other studies focused on predictors of FOSS projects sustainability and success. Some of the methods we have adopted for our study could be applied to similar research domains as for example the prediction of bug/issues in FOSS projects. A higher number of users and a growing number of revisions is intuitively correlated with the sustainability of the FOSS project. Further investigations are, however, neces-

sary to quantitatively confirm this correlation. We have built the two models by collecting data from the web pages and the source code versioning system of the SourceForge repository. Thousands of projects have been analysed and subsets of the projects were used to design two samples for modelling the prediction equations and two samples for validating the predictions. The download increment prediction equation achieved a 75 percent correctness of predictions and the new revisions contribution equation achieved an average of 93 percent of correct predictions. Both models have been tested according to guidelines and best practices available in the literature for developing new software measurements.

References

1. Kajan E (2002) Information technology encyclopedia and acronyms. Springer, Berlin Heidelberg New York
2. Broy M (2002) Software engineering – From auxiliary to key technologies. In: Broy M, Denert E (eds) Software Pioneers. Springer, Berlin Heidelberg New York
3. Che M, Grellmann W, Seidler S (1997) Appl Polym Sci 64:1079–1090
4. Ross DW (1977) Lysosomes and storage diseases. MA Thesis, Columbia University, New York
5. Taibi D., Lavazza L., Morasca S. (2007) OpenBQR: A framework for the assessment of OSS, Open Source Software 2007, (Limerick, Ireland, June 2007).
6. Petrinja E., Nambakam R., Sillitti A. (2009) Introducing the OpenSource Maturity Model. Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development collocated with 31st International Conference on Software Engineering (Vancouver, Canada, ICSE 2009), 37–41.
7. Petrinja E., Sillitti A., Succi G. (2010) Comparing OpenBRR, QSOS, and OMM Assessment Models. In the proceedings of the 6th International Conference on Open Source Systems (OSS 2010) (Notre Dame, USA, May 2010), 224–238.
8. Deprez J.-C., Simons A. (2008) Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR and QSOS, Book chapter in Lecture Notes in Computer Science, Ed. Springer Berlin, 189–203.
9. Stallman R. (1986) GNU's Bulletin, Volume 1, Number 1, page 8, URL http://www.gnu.org/bulletins/bull1.txt.
10. Alshayeb M., Li W. (2003) An Empirical Validation of Object-Oriented Metrics in Two Iterative Processes, IEEE Trans. Software Eng., vol. 29, no. 11, pp. 1043–1049, Nov. 2003.
11. Paulson J.W., Succi G., Eberlein A. (2004) An Empirical Study of Open-Source and Closed-Source Software Products, IEEE Trans. Software Eng., vol. 30, no. 4, pp. 246–256, Apr. 2004.
12. Scacchi W. (2004) Understanding free/open source software evolution. Software Evolution, Madhavji NH, Lehman MM, Ramil JF, Perry D (eds.). Wiley: New York NY, 2004.
13. Crowston K., Scozzi B. (2002) Open source software projects as virtual organizations: Competency rallying for software development. IEE ProceedingsSoftware Engineering 2002; 149(1):3-17.

14. Stewart K.J., Ammeter A.P., and Maruping L.M. (2006) Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects, J. Information Systems Research, vol. 17, no. 2, pp. 126–144, June 2006.
15. Capiluppi A., Lago P., Morisio M. (2003) Evidences in the evolution of OS projects through Changelog analyses. Proceedings 3rd Workshop on Open Source Software Engineering, 25th International Conference on Software Engineering, 19–24.
16. Koch S. (2007) Software Evolution in Open Source ProjectsA Large-Scale Investigation, J. Software Maintenance and Evolution: Research and Practice, vol. 19, no. 6, pp. 361-382.
17. Robles-Martinez G., Gonzalez-Barahona J.M., Centeno-Gonzalez J., Matellan-Olivera V., Rodero-Merino L. (2003) Studying the evolution of libre software projects using publicly available data. Proceedings 3rd Workshop on Open Source Software Engineering, 25th International Conference on Software Engineering, 2003; 111–116.
18. Crowston K., Annabi H., Howison J., Masango C. (2004) Towards a portfolio of FLOSS project success measures. Collaboration, Conflict and Control: The 4th Workshop on Open Source Software Engineering (ICSE 2004).
19. Crowston K., Annabi H., and Howison J. (2003) Defining Open Source Project Success, Proc. Intl Conf. Information Systems, 2003.
20. Grewal R., Lilien G.L., and Mallapragada G. (2006) Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems, J. Management Science, vol. 52, pp. 1043–1046.
21. Polancic G., Horvat R., and Rozman T. (2004) Comparative Assessment of Open Source Software Using Easy Accessible Data, Proc. 26th Intl Conf. Information Technology Interfaces, vol. 1, pp. 673–678.
22. Godfrey M.W., Tu Q. (2000) Evolution in open source software: A case study. Proceedings International Conference on Software Maintenance, 2000; 131–142.
23. Robles G., Gonzalez-Barahona J.M., Merelo J.J. (2006) Beyond source code: The importance of other artifacts in software development (a case study). Journal of Systems and Software 2006; 79(9):1233–1248.
24. Mishra A., Mishra D. (2006) Software quality assurance models in small and medium organisations: a comparison. IJITM 5(1): 4–20.
25. Raja U., Tretter M.J. (2012) Defining and Evaluating a Measure of Open Source Project Survivability. IEEE Trans. Software Eng. 38(1): 163–174.
26. McConnell S. (1999) Open-source methodology: Ready for prime time? IEEE Software 1999; 16(4):6-8.
27. Yiftachel P., Peled D., Hadar I., and Goldwasser D. (2006) Resource allocation among development phases: an economic approach. In Proceedings of the 2006 international workshop on Economics driven software engineering research (EDSER '06). ACM, New York, NY, USA, 43–48.
28. DeLone W.H., McLean E.R. (1992) Information Systems Success: The Quest for the Dependent Variable. Information Systems Research 3(1): 60–95.
29. Hippel E. von, Krogh G. von (2003) Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science. Organization Science, Vol. 14, No. 2, March-April 2003, pp. 209.