[Proceeding] An Empirical analysis of Open Source Software Defects data through Software Reliability Growth Models

An Empirical analysis of Open Source Software Defects data through Software Reliability Growth Models

Najeeb Ullah, Maurizio Morisio

Control and Computer Engineering Department, Politecnico Di Torino
10129, Torino, Italy

najeeb.ullah@polito.it, maurizio.morisio@polito.it

***Abstract:* — The purpose of this study is to analyze the reliability growth of Open Source Software (OSS) using Software Reliability Growth Models (SRGM). This study uses defects data of twenty five different releases of five OSS projects. For each release of the selected projects two types of datasets have been created; datasets developed with respect to defect creation date (created date DS) and datasets developed with respect to defect updated date (updated date DS). These defects datasets are modelled by eight SRGMs; Musa Okumoto, Inflection S-Shaped, Goel Okumoto, Delayed S-Shaped, Logistic, Gompertz, Yamada Exponential, and Generalized Goel Model. These models are chosen due to their widespread use in the literature. The SRGMs are fitted to both types of defects datasets of each project and the their fitting and prediction capabilities are analysed in order to study the OSS reliability growth with respect to defects creation and defects updating time because defect analysis can be used as a constructive reliability predictor. Results show that SRGMs fitting capabilities and prediction qualities directly increase when defects creation date is used for developing OSS defect datasets to characterize the reliability growth of OSS. Hence OSS reliability growth can be characterized with SRGM in a better way if the defect creation date is taken instead of defects updating (fixing) date while developing OSS defects datasets in their reliability modelling.**

***Keywords*— Software Reliability Growth Models, SRGM, Open Source Software, Empirical Study, Software Reliability Models, Open Source Defect Data**

I. INTRODUCTION

Open Source Software is software whose source code is freely accessible and changeable by the users, subject to constraints expressed in a number of licensing modes. It implies a global alliance for developing quality software with quick bug fixing along with quick addition and change in the software features on the end user's requirements basis. That is why OSS is fulfilling users' requirements very precisely to their choices and interests.

In the recent year tendency toward adoption of open source software and open source software components has swiftly increased. According to Gartner's report about 80% of the software will use open source technology by 2012 [1]. According to netcarf survey more than 58% web servers are using an open source web server, Apache [2]. The swift increase in the taking on of the open source technology is due to its freely availability, freedom of choice and affordability. There are still fears and unsolved questions especially for business people and project managers. Two common fears, which have also been outlined by Ray Lane, former Oracle executive in a keynote speaking in the open source conference 2004, are the lack of formal support and velocity of changes [2]. All these fears and concerns can be traced back to the quality and reliability of open source products.

Reliability is defined as the probability of failure free operation of software for specified period of time in a specified environment [3]. Reliability is one of the more important characteristics of software quality when considered for commercial use. Adoption of reliable open source products for commercial use can be a real challenge. While open source software products routinely provide information about product activity rank, number of developers and the number of users or downloads, this information does not convey information about the quality of the open source product.

Software reliability growth models (SRGMs) are frequently used in the literature for reliability characterization of commercial software. SRGM assume that reliability grows after a defect has been detected and fixed. However, results regarding the applicability of SRGMs for reliability characterization of OSS reported in the literature are not clear.

Here we characterize the reliability growth of OSS projects using SRGMs in order to investigate whether or not OSS reliability growths can be characterized through SRGM. If it does then the same tests of product reliability can be applied. If not, then new tests and models for analysing OSS reliability must be developed. In OSS projects defects detection and fixing time for a defect is quite different from each other. We also analyse the OSS reliability growth with respect to defects detection time and defects fixing time because defect analysis can be used as a constructive reliability predictor and measuring defect growth is a good empirical way of evaluating software quality [4].

We first provide a quick refresher on reliability modelling and describe common models used to measure software reliability in section 2. In section 3 we describe literature review. Then we describe the research questions and methodology that is used for this study in section 4. Section 5 describes data collection. In section 6 we describe the results and discuss the reliability growth for selected projects. Section 7 describes threats to validity of the study. Section 8 gives discussion on our findings and concludes the paper.

II. Software Reliability Growth Model

Software reliability modelling (SRM) has long been used as the most important and successful predictor of software quality when it hits the market. Reliability model is a mathematical expression that specifies general form of failure occurrence as a function of fault introduction, fault removal and operational environment [3]. Software Reliability Models (SRM) can both assess and predict reliability. In reliability assessment SRM are fitted to the collected failure data using statistical techniques (e.g. Linear Regression, Non Linear regression) based on the nature of collected data. In reliability prediction, the total number of expected future failures is forecasted on the basis of fitted SRM. Both assessment and prediction need good data, which implies accuracy i.e. data is accurately recorded at the time the failures occurred and pertinence i.e. data relates to an environment that resembles to the environment for which the forecast is performed [3]. For reliability modelling, software systems are tested in an environment that resembles to the operational environment. When a failure (i.e. an unexpected and incorrect behaviour of the system) occurs during testing, it is counted with a time tag. Cumulative failures are counted with corresponding cumulative time. SRM is fitted to the collected data and the fitted models are then used to predict the total number of expected defects (i.e. fault on the execution of which failure occur) in the software.

Hence, typically reliability modelling is composed of 5 steps: keeping a log of past failures, plotting the failures, determining a curve (i.e. Model) that best fits the observations, measuring how accurate the curve model is and then using the best fitted model predicting the future reliability in terms of predicting total number of expected defects in the software system.

The widely used SRM are Software Reliability Growth Models (SRGM). They assume that reliability grows after a defect has been detected and fixed. SRGM can be applied to guide the test board in their decision of whether to stop or continue the testing. These models are grouped into concave and S-Shaped models on the basis of assumption about failure occurrence pattern. The S-Shaped models assume that the occurrence pattern of cumulative number of failures is S-Shaped: initially the testers are not familiar with the product, then they become more familiar and hence there is a slow increase in fault removing. As the testers' skills improve the rate of uncovering defects increases quickly and then levels off as the residual errors become more difficult to remove. In the concave shaped models the increase in failure intensity reaches a peak before a decrease in failure pattern is observed. Therefore the concave models indicate that the failure intensity is expected to decrease exponentially after a peak was reached.

SRGMs measure and model the failure process itself. Because of this, they include a time component, which is characteristically based on recording times t_i of successive failures i ($i \geq 1$). Time may be recorded as execution time or calendar time. These models are fitted to the collected cumulative defects with respect to cumulative collected time. These fitted models then use to predict future behavior of the software. These models focus on the failure history of software. The failure history is affected by a number of factors, including the environment within which the software is executed and how it is executed. A general assumption of these models is that software must be executed according to its operational profile; that is, test inputs are selected according to the probability of their occurrence during actual operation of the software in a given environment [5]. There are many detailed descriptions of SRGM ([6], [7], [5], [8], [9], [10], [11]) with many studies and applications of the models in various contexts ([12], [13], [14]). Models differ based on their assumptions about the software and its execution environment.

In this research, we will plot defects data of OSS projects using eight SRGM models and examine the reliability growth in order to analyze the reliability pattern of OSSs. This study used eight SRGM, selected because they are the most representative in their category. Table 1 reports their name and reference and, for each of them:

- m (t) = mean value function (MVF) that represents the cumulative number of failures through time t

Each model has a different combination of parameters in the MVF:

- a = expected total number of defects in the code
- b = shape factor, i.e. the rates at which failure rate decreases
- c = expected number of residual faults in software at end of system test

Table 1: Summary of SRGM used in this study

Model Name	Type	Mean Value Function, m (t)
Musa-Okumoto [15]	Concave	$m(t) = a\ln(1+bt), \quad a>0, b>0$
Inflection S-Shaped [16]	S-Shaped	$m(t) = a\frac{1-\exp[-bt]}{1+\Psi(r)\exp[-bt]}, \Psi(r) = \frac{1-r}{r} a>0, b>0, r>0$
Goel-Okumoto [16]	Concave	$m(t) = a(1-\exp[-bt]), \quad a>0, b>0$
Delayed S-Shaped [16]	S-Shaped	$m(t) = a(1-(1+bt)\exp[-bt]), a>0, b>0$
Generalized Goel [16]	Concave	$m(t) = a(1-\exp[-bt^c]), a>0, b>0, c>0$
Gompertz [16]	S-Shaped	$m(t) = ak^{b^t}, a>0, 0<b<1, 0<k<1,$
Logistic [16]	S-Shaped	$m(t) = \frac{1}{1+k\exp[-bt]}, a>0, b>0, k>0,$
Yamada Exponential [17]	Concave	$m(t) = a\left(1-e^{-r\left(1-e^{-bt}\right)}\right), a>0, b>0, r>0.$

III. LITERATURE REVIEW

Different studies are available in the literature about the applicability of software reliability models for OSS, with unclear results. Syed Mohammad et al. [18] examined the defect discovery rate of two OSS products with software developed in-house using 2 SRGM. They observed that the two OSS products have a different profile of defect discovery. Ying Zhou et al [19] analysed bug tracking data of 6 OSS projects. They observed that along their developmental cycle, OSS projects exhibit similar reliability growth pattern with that of closed source projects. They proposed the general Weibull distribution to model the failure occurrence pattern of OSS projects. Bruno Rossi et al [20] analysed the failure occurrence pattern of 3 OSS products applying SRGM. They proposed that the best model for OSS is the Weibull distribution. Cobra Rahmani et al. [21] compared the fitting and prediction capabilities of 3 models using failure data of 5 OSS projects. They observed Shneidewind model is the best while Weibull is the worst one. Fengzhong et al [22] examined the bug reports of 6 OSS projects. They modelled the bug reports using nonparametric techniques. They suggested that Generalized Additive (GA) models and exponential smoothing approaches are suitable for reliability characterization of OSS projects.

Hence in a generalized way empirical validation of software reliability models for OSS projects is needed, in order to make clear the applicability of software reliability models for OSS projects.

IV. GOALS, RESEARCH QUESTIONS AND METRICS

The existing body of the literature on reliability characterization of OSS through SRGMs is limited. Further, the results regarding the applicability of SRGMs for reliability characterization of OSS reported in the literature are not clear. That is why the first goal of this study is to empirically investigate whether or not OSS reliability growths can be characterized through SRGM. Secondly in OSS projects defects detection and fixing time for a defect is quite different from each other, which may affect the reliability growth. This difference in defect detecting and fixing time may be a reason of unclear results reported in literature regarding the applicability of SRGM for reliability characterization of OSS. We therefore empirically analyse the reliability growth of OSS with respect to defect detecting time versus defect fixing time through SRGM.

To achieve the goals, our study focuses on these research questions, which are presented in detail:

R.Q1: Are SRGM models' fitting capabilities affected by defect detection and fixing time?

Or, in operational terms, the OSS defect occurrence trend can be represented by SRGM in a similar way such like OSS defect fixing trend? Models are fitted to the defects datasets collected with respect to defect detection date (DD DS) and to the defects datasets collected with respect to defect fixing date (DF DS), and their R^2 are analysed and compared by adopting visual analysis and statistical hypothesis testing. Model fitting is required to estimate the parameters of the models and produce a prediction of failures. Fitting can be done using Linear or Non Linear Regression (NLR). In linear regression, a line is determined that fit to data, while NLR is a general technique to fit a curve through data. The parameters are estimated by minimizing the sum of the squares of the distances between data points and the regression curve. We will use NLR fitting due to the nature of data.

NLR is an iterative process that starts with initial estimated values for each parameter. The iterative algorithm then gradually adjusts these until to converge on the best fit so that the adjustments make virtually no difference in the sum-of-squares. A model's parameters do not converge to best fit if the model cannot describe the data. On consequence the model cannot fit to the data. We use a commercial program for curve fitting of the models to the collected defect datasets. In case of convergence of the curve fitting we use goodness of fit (GOF) test, R^2 [23] to determine how well curve fit to the data. It is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{k}(m_i - m(t_i))^2}{\sum_{i=1}^{k}\left(m_i - \sum_{j=1}^{k}\frac{m_j}{n}\right)^2}$$

In the expression k represents the size of the data set, $m(t_i)$ represents predicted cumulative failures, m_i represents actual cumulative failures at time t_i and n represents number of data points in the dataset. R^2 takes a value between 0 and 1, inclusive. The closer the R^2 value is to one, the better the fit. The R^2-value is used for its simplicity and is motivated by the work of Gaudoin, O. et al [24], who evaluated the power of several statistical tests for GOF for a variety of reliability models. Their evaluation showed that this measure was as least as powerful as the other GOF tests analysed.

For the purpose of visual representation, we use box plots: as they allow for an immediate comparison. We consider a good fit when $R^2 > 0.90$ because the model fit might be considered good having $R^2 = 0.90$. This threshold categorizes the models as good and bad in term of fitting capability. We also do hypothesis testing on the R^2 of fitted models in order to determine statistical significant difference in models fitting values for both types of datasets. Therefore we formulate null and alternative hypothesis as follows.

$H0_0$: The SRGM models' fitting capabilities for OSS are not affected with defect detecting and fixing time (i.e. R^2 of models fitted to DD DS is not different from R^2 of models fitted to DF DS).

H0a: The SRGM models' fitting capabilities for OSS are affected with defect detecting and fixing time (i.e. R^2 of models fitted to DD DS is better than R^2 of models fitted to DF DS).

According to the recommendations in [25] we use the Mann-Whitney test in order to evaluate practical differences in models' fitting capabilities for both types of datasets. The assumption to select was the not normal distribution of datasets comprising of R^2 values of fitted models. In the statistical testing, the significance level is checked by the

given p-value. For rejecting or accepting the null hypothesis, we used the significance value α=5%.

R.Q2: Are SRGM models' predictive qualities affected by defect detection and fixing time?

Or in operational terms, the models prediction accuracy and correctness do not change with respect to defect detection and fixing time. We use the partial failure history (i.e. first portion of the collected defect datasets are used for model fitting and remaining portion of the datasets are used for prediction) of the products to accomplish the prediction as [26]. The first two thirds data points of the each datasets following [27], is used to estimate the parameters. These estimated values of the parameters are then applied to the entire time span for which failure data is collected in each dataset in order to compare the prediction qualities of the models for both types of defect datasets.

Prediction capability can be evaluated under two points of view, accuracy and correctness. Accuracy deals with the difference between estimated and actual over a time period. Correctness deals with the difference between predicted and actual at a specific point in time (e.g. release date). A model can be accurate but not correct and vice versa. For this reason we use the Theil's Statistic (TS) for accuracy and Predicted Relative Error (PRE) for correctness.

1) The Theil's statistic (TS) is the average deviation percentage over all data points. The closer Theil's statistic is to zero, the better the prediction accuracy of the model. It is defined as [28]:

$$TS = \sqrt{\frac{\sum_{i=1}^{k}(m(t_i)-m_i)^2}{\sum_{i=i}^{k} {m_i}^2}} * 100\%$$

2) Predicted Relative Error is a ratio between the error difference (actual versus predicted) and the predicted number of defects at the time point of failures prediction (e.g. release time).

$$\text{PRE} = \frac{\text{Predicted} - \text{Actual No of Defects}}{\text{Predicted}}$$

Similar to models fitting (i.e. R^2), models prediction accuracy and correctness are visually represented through boxplots. We consider a prediction as good if TS is below 10% and PRE is within the range [-10%, +10%] of total number of actual defects because 10% range might be acceptable. These thresholds of TS and PRE categorize the models as good and bad in term of prediction accuracy and correctness. We also do hypothesis testing on the TS and PRE of fitted models in order to determine statistical significant difference in models prediction qualities for both types of datasets. Therefore we formulate null and alternative hypotheses as follows.

$H1_0$: The SRGM models' prediction qualities for OSS are not affected with defect detecting and fixing time (i.e. TS and PRE of models prediction for DD DS is not different from TS and PRE of models prediction for DF DS).

H1a: The SRGM models' prediction qualities for OSS are affected with defect detecting and fixing time (i.e. TS and PRE of models prediction for DD DS is better than TS and PRE of models prediction for DF DS).

Similar to methodology adopted for RQ1, we use the Mann-Whitney test in order to evaluate practical differences in models' prediction qualities for both types of datasets. The assumption to select was the not normal distribution of datasets comprising of TS and PRE values of fitted models. In the statistical testing, the significance level is checked by the given p-value. For rejecting or accepting the null hypothesis, we used the significance value α=5%.

V. DATA COLLECTION

In OSS projects defects detection and fixing time for a defect is quite different from each other. The goal of the study is to analyse the reliability growth of OSS with respect to defect detection time versus defect fixing time. We identified five notable and active open source projects from apache.org (https://issues.apache.org/). These projects are C++ Standard Library, JUDDI, HTTP Server, XML Beans, and Enterprise Social Messaging Environment (ESME). The Apache C++ Standard Library provides a free implementation of the ISO/IEC 14882 international standard for C++ that enables source code portability and consistent behaviour of programs across all major hardware implementations, operating systems, and compilers, open source and commercial alike. JUDDI is an open source Java implementation of the Universal Description, Discovery, and Integration (UDDI v3) specification for (Web) Services. The Apache HTTP Server is an open-source HTTP server for modern operating systems including UNIX, Microsoft Windows, Mac OS/X and Netware. XML Beans is a tool that allows you to access the full power of XML in a Java friendly way. ESME (Enterprise Social Messaging Environment) is a secure and highly scalable micro sharing and micro messaging platform that allows people to discover and meet one another and get controlled access to other sources of information, all in a business process context. All these projects are considered stable in production. The 66%, 95%, 68%, 64% and 82% of the reported issues in these projects respectively, have been fixed and closed. We collected defect data of the selected projects from apache.org using JIRA. JIRA is a commercial issue tracker. Issues can be bugs, feature requests, improvements, or tasks. JIRA track bugs and tasks, link issues to related source code, plan agile development, monitor activity, report on project status.

For each release of the selected projects we have collected all the issues reported at our date of observation. For each project, we have considered all the major releases until October 2012. We were able to get eight (8) versions for C++ Standard Library, seven (7) versions for JUDDI, two (2) versions for HTTP Server, five (5) versions for XML Beans and three (3) versions for ESME. Hence defects data of 25 different releases of 5 projects were collected. Table 2 lists the

Table2: Selected Projects Details

Project	Version	Release Date
C++ Standard Library	V4.1.2	18/07/2005
	V4.1.3	30/01/2006
	V4.1.4	03/07/2006
	V4.2.0	29/10/2007
	V4.2.1	01/05/2008
	V4.2.2	30/06/2008
	V4.2.3	01/09/2008
	V5.0.0	31/05/2009
JUDDI	V2.0	02/08/2009
	V3.0	26/10/2009
	V3.0.1	01/02/2010
	V3.0.2	17/05/2010
	V3.0.3	22/07/2010
	V3.0.4	06/11/2010
	V3.1.0	27/06/2011
HTTP Server	V3.1.4	13/02/2005
	V3.2.7	13/02/2006
XMLBeans	V2.0	30/06/2005
	V2.1	16/11/2005
	V2.2	23/03/2006
	V2.3	01/06/2007
	V2.4	08/07/2008
ESME	V1.1	09/10/2010
	V1.2	14/03/2011
	V1.3	29/08/2011

information of the projects along with the selected releases and their time windows for each release.

The tracking software records all the information regarding each issue, such as *issue type*, *status*, *created date*, *updated date*, *affected version*. After a deep inspection of the repositories and of their documentation, we have decided to focus on those issues that were declared "bug" or "defect" excluding "enhancement," "feature-request," "task" or "patch". For the same reason, we have considered only those issues that were reported as closed or resolved after the release date of each version. Further, we excluded issues closed before the release date. These issues are typically found in the candidate (or testing) releases of projects. We filtered all the issues in order to collect only issues that have declared "defect" or "bug" as in [20, 22]. For the filtration of the collected issue from the online repository we used the aforementioned attributes. After refining the data we grouped the defects into cumulative defects by week.

We developed two types of datasets for each release of each project. In first type of datasets (i.e. created date DS) we grouped the defects into cumulative defects by weeks with respect to created date of the defects while in second type of dataset (i.e. updated date DS) we grouped the defects into cumulative defects by weeks with respect to updated date of the defects. We divided the entire time span of each release into weeks and then counted detected defects in each week. For each release in first type of dataset we counted defects for each week with respect to created date (after this will call created date DS) and in second type of dataset we counted defects for each week with respect to updated date (after this will call updated date DS). In this way we developed 25 created date DS and 25 updated date DS for total of 25 selected releases of the five OSS projects. The complete datasets are available online[1].

VI. RESULTS

A: Models Fitting Results: (RQ1)

In Figure 1 we report the boxplots of R^2 (i.e. Goodness of Fit values) per model for both types of datasets of each release of the selected projects. For RQ1, observing the box plots in Figure 1 it appears that there is clear difference. Medians of all the models are above the threshold in case of created date DS and all the models have also narrow boxplot (always better than 0.9, the threshold depicted as a red horizontal line) but some outliers. On contrary in case of updated date DS the boxplots of R^2 values show clear variation. It is clear from the Figure 1 that models fitting capabilities increase in case of created date DS. Hence it is suggested that for the reliability characterization of OSS through SRGMs defects created date should be considered.

We also test the hypothesis $H0_0$ with Mann-Whitney test for differences. The test reports a p-value = 0.0006344 which is below the threshold, α. Therefore, we reject the null hypothesis, indicating that there is significant difference of models fitting between the defects created and updated date DS, which is also visually represented through boxplots in Figure 1.

In summary:

- All the models have very good fit (better than 0.9), but with outliers in the case of created date DS while in updated date DS only the median of Inflection, Logistic, Gompertz and Generalized are above the threshold.
- There is practical significant difference in models fitting capabilities when defects created date is used in developing OSS defects datasets for the reliability characterization.

B. Models Prediction Results: (RQ2)

In order to analyse the models prediction qualities we used the first two-third data points of the data sets to train the model, and predicted the last third. The choice of two-third data points was motivated with the wood's suggestion for model stability [14]. We analyse the models prediction qualities in terms of prediction accuracy and correctness.

Accuracy

Figure 2 reports the TS values for all datasets of both types. The red line represents the 0.1 threshold, usually considered indicator of good accuracy.

In created date DS, all the models have very good prediction accuracy and have narrow boxplot (always the medians lie on the threshold 0.1, the threshold depicted as a red horizontal line). In updated date DS all the models have not good prediction accuracy and the boxplots show the variations in their prediction accuracy. It is clear from the Figure 2 that models prediction accuracy increase in case of created date

[1] http://softeng.polito.it/najeeb/DataSets/OSSDS.pdf

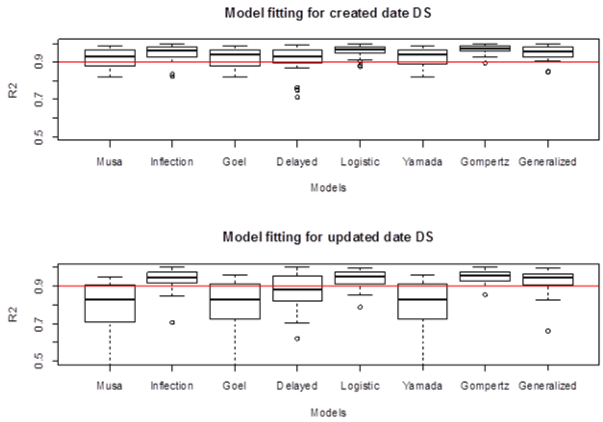

Figure 1: Box Plots of fitting (R^2) values

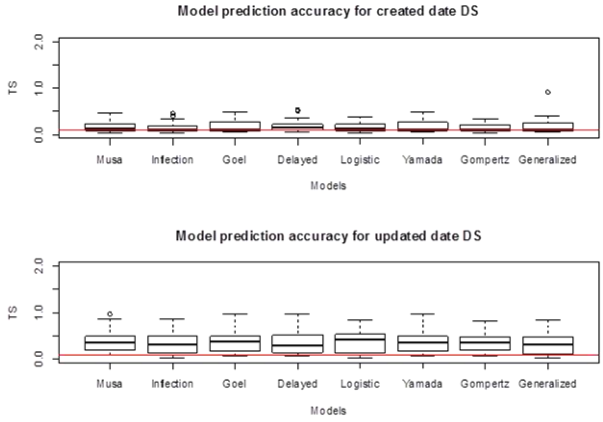

Figure 2: Box Plots of Prediction Accuracy (TS) values

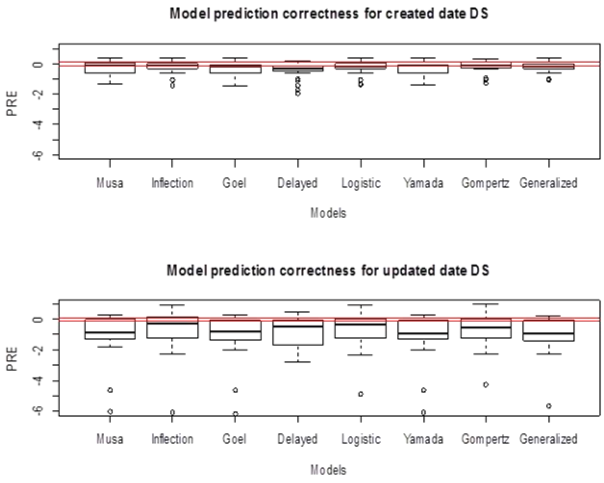

Figure 3: Box Plots of Prediction Correctness (PRE) values

DS. Hence it is suggested that for the reliability characterization of OSS through SRGMs defects created date should be considered.

We test the hypothesis $H1_0$ with Mann-Whitney test for differences. The test reports a p-value < 2.2e-16 for TS values of both types of datasets, which is below the threshold, α. Therefore, we reject the null hypothesis, indicating that there is significant difference of models prediction qualities in term of prediction accuracy between the defects created and updated date DS, which is also visually represented through boxplots in Figure 2.

In summary:

- In created date DS all models are close to the threshold while in updated date DS all other models have variations.
- There is practical significant difference in models prediction accuracy when defects created date is used in developing OSS defects dataset for the reliability characterization.

Correctness

Correctness results are shown in the boxplots of Figure 3. The red lines represent the range ±10% of total number of actual defects.

In created date DS all models have narrow boxplots and their medians lie within the range ±10% of selected threshold. On contrary in updated date DS all the models tend to underestimate the actual number of defects. Only inflection S-Shaped has median lies in the selected range. It is clear from the Figure 3 that models prediction correctness increase in case of created date DS. Hence it is suggested that for the reliability characterization of OSS through SRGMs defects created date should be considered.

We test the hypothesis $H1_0$ with Mann-Whitney test for differences. The test reports a p-value = 9.709e-05 for PRE values of both types of datasets, which is below the threshold, α. Therefore, we reject the null hypothesis, indicating that there is significant difference of models prediction qualities in term of prediction correctness between the defects created and updated date DS, which is also visually represented through boxplots in Figure 3.

In summary:

- On created date DS all the models provide good accuracy and prediction while for updated date DS all the models behave inversely.
- There is practical significant difference in models prediction accuracy when defects created date is used in developing OSS defects dataset for the reliability characterization.

VII. THREATS TO VALIDITY

We recognize a first conclusion threat is the choice of threshold is not grounded in the literature. However we provided boxplots to show to the readers that certain models got good fitting\prediction performances in several datasets. Although the high number of datasets used (50) might make our findings generalizable, we strongly suggest the reader to define her own thresholds for fitting, accuracy and correctness of predictions and re elaborate the results according to those thresholds, using the boxplot provided. We notice another conclusion threat in the choice of not performing cross validation in prediction. However we grounded our choice in the literature.

The number of release and the time windows of the observations are different in the five OSS. This was due to

some time constraints and the availability of the data in the repositories. As we do not compare the five OSS, but we rather want to understand whether there is a pattern of reliability in each OSS, this difference is not crucial.

We used open on-line repository to collect data of five different projects. We intensively cleaned the data we collected to limit the bias associated with the open nature of these repositories.

VIII. DISCUSSION AND CONCLUSION

We have attempted to derive general conclusion about the reliability growth of OSS applying eight different SRGM models to a wide range of OSS defects datasets. We evaluate the reliability growth pattern of OSS using SRGMs. The performance of models differs between created date and updated date datasets. The results show a huge difference between failures occurrence patterns of created date DS and updated date DS, which indicates a clear cut difference in the reliability growth of the OSS with respect to defect creating date and defect fixing date. From the results of this study it is suggested that for reliability characterization of OSS defects created date should be considered because the reliability of OSS directly increases with defects created date. The results also show that SRGM can be used for the reliability characterization OSS and their fitting capabilities and prediction qualities directly related to defects creating date instead of defects updating/fixing date. This study makes the unclear results reported in the literature regarding the applicability of SRGMs for OSS reliability characterization, clearer.

In our previous studies [29, 30] we have observed different behaviour of the best models for OSS as compared to CSS (Closed Source Software). The best performer models were Musa Okumoto and Inflection for industrial datasets, while Gompertz and Inflection were the best for OSS datasets. We therefore deeply investigate the models fitting and prediction results focusing on this observation. We observed that all the S-Shaped models fitting and prediction qualities for OSS is better than concave shaped that is why Musa is best former for CSS but not for OSS because of its concave nature. While Gompertz and Inflection belong to S-Shaped category and as such it indicates an initial learning phase in which the community of end-users and reviewers of the open source project does not react promptly to new release. So because of this S-Shaped nature Inflection S-Shaped and Gompertz outperformed for OSS than Musa Okumoto Model.

These results of this study show that SRGM models can characterize the OSS reliability growth. For reliability modelling of OSS the defect creating date should be used for developing defect data sets of OSS in order to characterize their reliability growth through software reliability models.

REFERENCES:

[1] Najeeb ullah, Maurizio Morisio, "An Empirical Study of Open Source Software Reliability Models", Proceedings of International conference on computational intelligence and computing research, 2011.

[2] Farber D. Six barriers to open source adoption, ZDNet Tech Update, March, 2004 [WWW document].

[3] IEEE Std 1633-2008 IEEE Recommended Practice in Software Reliability.

[4] David, N. C. (2002) Managing Software Quality with Defects, In Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Redevelopment IEEE Computer Society.

[5] Lyu, M. (ed.): 1996, Handbook of Software Reliability Engineering. New York: McGraw-Hill.

[6] Goel, A. L., and Okumoto, K. 1979. "A time dependent error detection model for software reliability and other performance measures", IEEE Transactions on Reliability 28(3): 206–211.

[7] Kececioglu, D. 1991. Reliability Engineering Handbook, Vol. 2, Englewood Cliffs, NJ: Prentice-Hall.

[8] Musa, J., Iannino, A., and Okumoto, K. 1987. Software Reliability: Measurement, Prediction, Application. New York: McGraw-Hill.

[9] Trachtenberg, M. 1990, "A general theory of software-reliability modeling", IEEE Transactions on Reliability 39(1): 92–96.

[10] Yamada, S., Ohba, M., and Osaki, S. 1983. "S-shaped reliability growth modeling for software error detection", IEEE Transactions on Reliability 32(5): 475–478.

[11] Yamada, S., Ohtera, H., and Narihisa, H. 1986. Software reliability growth models with testing effort. IEEE Transactions on Reliability 35(1): 19–23.

[12] Musa, J., and Ackerman, A. 1989, Quantifying software validation: When to stop testing. IEEE Software. 19–27.

[13] Wood, A. 1996, Predicting software reliability. IEEE Computer 29(11): 69–78.

[14] Wood, A. 1997. Software reliability growth models: Assumptions vs. reality. Proceedings of the International Symposium on Software Reliability Engineering 23(11): 136–141.

[15] J. D. Musa and K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement," in *Conf. Proc. 7th International Conf. on Softw. Engineering*, 1983, pp. 230–237.

[16] C. Y. Huang, M. R. Lyu, and S. Y. Kuo, "A unified scheme of some non-homogenous Poisson process models for software reliability estimation," *IEEE Trans. Softw. Engineering*, vol. 29, no. 3, pp. 261–269, March 2003.

[17] Yamada, S., Ohtera, H., and Narihisa, H. 1986. Software reliability growth models with testing effort. IEEE Transactions on Reliability 35(1): 19–23.

[18] Syed-Mohamad et al, "Reliability Growth of Open Source Software Using Defect Analysis", International Conference on Computer Science and Software Engineering, 2008.

[19] Ying Zhou et al, "Open source software reliability model: an empirical approach", ACM SIGSOFT Software Engineering Notes, 2005

[20] Bruno Rossi et al, "Modelling Failures Occurrences of Open Source Software with Reliability Growth", journal of Open Source Software: New Horizons, page 268-280, 2010.

[21] Cobra Rahmani et al, "A Comparative Analysis of Open Source Software Reliability", Journal of Software, page 1384-1394, 2010.

[22] Fenhzong et al, "Analyzing and Modeling Open Source Software Bug Report Data", 19th Australian Conference on Software Engineering.

[23] K. C. Chiu, Y. S. Huang, and T. Z. Lee, "A study of software reliability growth from the perspective of learning effects," Reliability Engineering and System Safety, pp. 1410–1421, 2008.

[24] O. Gaudoin et al, "A simple goodness-of- fit test for the power law process based on the Duane plot", IEEE Transactions on Reliability.

[25] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in software engineering: an introduction.* Norwell, Massachusetts. USA: Kluwer Academic Publishers, 2000.

[26] Cobra Rahmani et al, "A Comparative Analysis of Open Source Software Reliability", Journal of Software, page 1384-1394, 2010.

[27] Carina Andersson , "A replicated empirical study of a selection method for software reliability growth models", journal of Empirical Software Engineering, pages 161-182, year 2006.

[28] P. L. Li, J. Herbsleb, and M. Shaw, "Forecasting field defect rates using a combined time-based and metrics-based approach: a case study of OpenBSD," in Proceedings of the 16th IEEE International Symposium on Softw. Reliability Engineering, Chicago, IL, 2005, pp. 193–202.

[29] Najeeb Ullah, Maurizio Morisio, Antonio Vetro, "A Comparative Analysis of Software Reliability Growth Models using defect data of closed and open source software", proceedings of SEW-35, 2012.

[30] Najeeb Ullah, Maurizio Morisio, "An Empirical Study of reliability growth of open versus closed source software through software reliability growth models", proceedings of APSEC 2012.