A Comparative Analysis of Software Reliability Growth Models using defects data of Closed and Open Source Software

Najeeb Ullah, Maurizio Morisio, Antonio Vetro

Control and Computer Engineering Department Politecnico, Torino, Italy (surname@polito.it)

***Abstract*—The purpose of this study is to compare the fitting (goodness of fit) and prediction capability of eight Software Reliability Growth Models (SRGM) using fifty different failure Data sets. These data sets contain defect data collected from system test phase, operational phase (field defects) and Open Source Software (OSS) projects. The failure data are modelled by eight SRGM (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 prevalence among many software reliability models. The results can be summarized as follows**

• Fitting capability: Musa Okumoto fits all data sets, but all models fit all the OSS datasets.

• Prediction capability: Musa Okumoto, Inflection S-Shaped and Goel Okumoto are the best predictors for industrial data sets, Gompertz and Yamada are the best predictors for OSS data sets.

• Fitting and prediction capability: Musa Okumoto and Inflection are the best performers on industrial datasets. However this happens only on slightly more than 50% of the datasets. Gompertz and Inflection are the best performers for all OSS datasets.

***Index Terms*—Software Reliability Growth Models, SRGM, Open Source Software, Failure Data, Software Reliability Models**

I. Introduction

Software development is a brain intensive activity. Therefore, the quality of the product is subject to large variations. Reliability is one of the most important attributes of software quality, which is defined as the probability of failure-free software operation for a specified period of time in a specified environment [28]. Starting from the 70's different Software Reliability Models (SRM) have been proposed for software reliability characterization and prediction. SRM is a mathematical expression that specifies the general form of the software failure process as a function of factors such as fault introduction, fault removal, and the operational environment [28]. SRM is composed of different parameters. Parameter is a variable or arbitrary constant appearing in a mathematical expression, each value of which restricts or determines the specific form of the expression. The failure rate (failures per unit time) of a software system is generally decreasing due to fault identification and removal. Software Reliability modeling is done to estimate the form of the curve of the failure rate by statistically estimating the parameters associated with the selected model. The purpose of this measure is twofold: 1) to estimate the extra execution time during test required to meet a specified reliability objective and 2) to identify the expected reliability of the software when the product is released [28]. In general SRM are categorized as white box and black box. White box approaches analyze the structure i.e. the architecture of the software that has been specified and designed. These models predict the reliability of software on the basis of the relationship among different components and their interactions. These approaches are also called deterministic approaches. They are based on logical complexity, decision point, program length, operands and operators of software. Path-Based Models and State-Based Models are two examples of this type of reliability model. In the literature these models are known as Architecture Based Reliability Models. The Black Box approaches are classified into different types, Early Prediction Models, SRGM, Input Domain Based Model, and Hybrid Black Box Models. Our work is focused on SRGM models because of their widespread use. SRGM can be applied to guide the test board in their decision of whether to stop or continue the testing. Herein we present a comparative analysis of SRGM models in term of goodness-of-fit, prediction accuracy and correctness based on thirty eight failure data sets containing system test failures data, field and OSS defects data. The rest of the paper is organized as follows. Section 2 contains background information and literature review. Section 3 provides the goals and research questions of this study; section 4 describes models and data selection. Section 5 describes results; section 6 concludes the study and gives future research direction.

II. BACKGROUND

A. *Software Reliability Growth Models*

SRGM is one of the prominent classes of black box SRM. 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

1550-6215/13 $26.00 © 2013 IEEE
DOI 10.1109/SEW.2012.26

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 pick was reached. Software Reliability Growth Models measure and model the failure process itself. Because of this, they include a time component, which is characteristically based on recording times ti of successive failures i (i $\geq$1). Time may be recorded as execution time or calendar time. 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 [3]. There are many detailed descriptions of SRGM ([1], [2], [3], [4], [5], [6], [7]) with many studies and applications of the models in various contexts ([8], [9], [10]). Models differ based on their assumptions about the software and its execution environment.

B. Model Selection

Over the past 40 years many SRGM have been proposed for software reliability characterization. The recurring question is therefore which model to choose in a given context. Different models must be evaluated, compared and then the best one should be chosen [11]. Many researchers like Musa et al. [12] have shown that some families of models have certain characteristics that are considered better than others; for example, the geometric family of models (i.e. models based on the hyper-geometric distribution for estimating the number of residual software faults) has a better prediction quality than the other models. By comparison with different models, Schick and Wolverton [13], and Sukert [14], proposed a new approach, which suggested techniques for finding the best model for each individual application among the existing models. Brocklehurst et al. [15] proposed that the nature of software failures makes the model selection process in general a difficult task. They observed that hidden design flaws are the main causes of software failures. Goel's [16] paper stated that different models predict well only on certain data sets; and the best model for a given application can be selected by comparing the predictive quality of different models. Abdel-Ghaly et al. [17] analyzed the predictive quality of 10 models using 5 methods of evaluation. They observed that different methods of model evaluation select different model as best predictor. Khoshgoftaar [18] suggested Akaike Information Criteria (AIC), best model selection criteria. Khoshgoftaar and Woodcock [19] proposed a method for the selection of a reliability model among various alternatives using the log-likelihood function (i.e. a function of the parameters of the models). They applied the method to the failure logs of a project. In spite of the fact that many studies have been conducted, there is no agreement on how to select the best model before starting a project.

C. SRGM in Open Source System

Different studies are available in the literature about the applicability of software reliability models for OSS, with unclear results. Syed Mohamad et al. [23] 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 [24] analyzed 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 [25] analyzed 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. [26] 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 [27] 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.

III. GOALS, RESEARCH QUESTIONS AND METRICS

There is no agreement on what is the best reliability model for a given project, especially at its inception. Different models predict well only on certain data sets and the best model can be selected by comparing the predictive qualities of a number of models only at the end of a project. That is why the goal of this study is to compare the reliability characterization and prediction quality of different SRGM in order to draw a general conclusion about the best fitting and best predictor models among them. Moreover, we want to study SRGM models with both industrial and open source failure datasets in order to make clear ambiguous literature results in a generalized way. Herein, we summarize the goal and introduce the research questions that derive this study using the GQM [29] template.

Object of the study	Analyze different SRGM models
Purpose	to compare
Focus	Their capability to characterize and predict the reliability of a project
Stakeholder	from the point of view of maintenance and quality managers
Context factors	in the context of industrial and open source systems

We describe the research questions and metrics that complete the GQM. The first step is analyzing the capability of models to simply fit the data sets. At this regard we define RQ1 and compare the fitting capability, in terms of R^2, of the models on the whole dataset. The second step is analyzing the capability of prediction. To the regard of prediction we have two different RQs. RQ2 simply compares the models in terms of PRE and TS. RQ3 tries to help in selecting a model, taking the point of view of a project manager who only has available part of the dataset and needs to select a model for prediction. So RQ3 analyzes if a model with a good fit (high R^2) is also a good predictor. The RQs are now presented in detail.

RQ 1: Which SRGM models fit best?

Or, in operational terms, which SRGM has the best R^2? Models are fitted on the whole data sets, and their R^2 are analysed and compared. Model fitting is required to estimate the parameters of the models. Fitting can be done using Linear or Non Linear Regression (NLR) depending upon nature of the data. 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. On the contrary, in case of convergence of the iterative algorithm, the R^2 [20] is the metric that indicates how successful the fit is. We use R^2 for goodness of fit test because it is the more powerful measure [30]. It is defined as:

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

In the expression k represents the size of the data set, m(ti) represents predicted cumulative failures and mi represents actual cumulative failures at time ti. R^2 takes a value between 0 and 1, inclusive. The closer the R^2 value is to one, the better the fit. We consider a good fit when $R^2 > 0.90$. We analyse and rank models based on their R^2.

RQ 2: Which SRGM models are good predictors?

We use the partial failure history of the products to accomplish the prediction as [26]. The first two thirds data points of the each datasets following [22], 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. 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 [21]

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

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).

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

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. We rank and analyze models prediction based on their TS and PRE.

RQ 3: A model with good fit is also a good predictor?

Or, in operational terms, a model with a good R^2 also has good TS and PRE? Models are fitted on two thirds of the data sets, the R^2 is computed on this fit, TS and PRE are computed on the remaining third of the dataset. RQ2 tries to understand what models are best predictors, but takes an a posteriori view. RQ3 takes an a priori view, or uses the view of a project manager who has to decide what model to use before the end of the project (at two thirds of it), and only has the goodness of fit as a rationale for a decision.

IV. MODELS AND DATA SELECTION

A. SRGM models

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, mean value function, m (t) that represents the cumulative number of failures through time t. Each model has a different combination of parameters.

- 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 I
SUMMARY OF SRGM USED IN THIS STUDY

Model Name	Type	Mean Value Function, m (t)
Musa-Okumoto [31]	Concave	m(t) = a ln(1+bt)
Inflection S-Shaped [32]	S-Shaped	m(t) = a(1-e^{-bt})/(1+beta.e^{-bt})
Goel-Okumoto [32]	Concave	m(t) = a(1-e^{-bt})
Delayed S-Shaped [32]	S-Shaped	m(t) = a(1-(1+bt)e^{-bt})
Generalized Goel [32]	Concave	m(t) = a(1-e$^{-bt^c}$)
Gompertz [32]	S-Shaped	m(t) = ak$^{b^t}$
Logistic [32]	S-Shaped	m(t) = a/(1+ke^{-bt})
Yamada Exponential [7]	Concave	m(t) = a(1-e$^{-r.alpha(1-exp(-beta.t))}$)

B. Datasets

The goal of the study is to analyse the selected SRGM using as many software failure data sets as possible. For this purpose we collected failure data from the literature. We have searched papers on IEEE Explorer, ACM Digital Library and in three journals, i.e. Journal of Information and Software Technology, the Journal of System and Software and IEEE software. For papers searching these strings have been used: Software failure rate, Software failure intensity, Software failure Dataset, Failure rate and Reliability, Failure intensity and Reliability We found 2100 papers, 19 of which were relevant for our study because they contained failure data sets on 38 projects. Among these, 32 projects were closed source and 6 were Open Source. In 32 closed source projects, 22 contain system test failure data and 10 contain field defect data collected from the operation phase. OSS projects data have no distinction between phases. Due to space limitation the complete data sets along with their references are available online (http://softeng.polito.it/najeeb/DataSets/DS1.pdf). We have selected both system test and field defect data sets in order to evaluate the best fitting and best predictor SRGM for both system test and operation phase, because the software reliability models are used for the prediction of failure in both phases and the phase may be a factor for model selection. Six data sets contain defect data of two OSS Projects, Apache and GNOME. Three data sets have been collected from different versions of each of the two OSS projects, due to space limitation available online (http://softeng.polito.it/najeeb/DataSets/DS1.pdf). Apart from this we identified two notable and active open source projects from apache.org (https://issues.apache.org/). Both of the projects, C++ Standard Library and JUDDI are considered stable in production. The 66% of the reported issues in first project have been fixed while in the second project 95% of the reported issues have 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. For each version we have collected all the issues reported at date of observation together with the date at which they were reported (date of opening). For each open source project, we have considered all the major versions until April 2012. For C++ Standard Library we were able to get eight (8) versions. Unfortunately, JUDDI had not so many reports and versions as compared to C++ Standard Library and we had to limit the versions to four (4) for JUDDI until October 2011. we have focused 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. The complete datasets due to space limitation are available online (http://softeng.polito.it/najeeb/confdata/DS2.pdf).

V. Results

A. *RQ1: Which Model fit best?*

First we consider the basic capability of a model to fit the dataset (fits or not), irrespective of the goodness of fit (R^2). Fig. 1 reports for each model on the X axis the percentage of datasets fitted (axis Y) in each data group (colour bars). Musa fitted to all data sets in each data group, most models also fit, except Yamada exponential, Gompertz, Generalized Goel that fit poorly, especially field test and system test datasets. In order to analyze goodness of fit, Fig. 2 reports how many times a model is the one with best R^2. For instance Musa Okumoto has the best R^2 on 60% of field defect datasets. Musa Okumoto is top performer on field defects data; Gompertz has very good results on OSS datasets directly followed by Inflection S-Shaped Model. But apart from that there is no clear best model. However, analyzing the top performer only as in Fig 2 can be misleading, in case many models fit with a similar R^2 the same dataset. Therefore in boxplots of Figure 5 we report the boxplots of R^2 per model and per dataset category. It should be reminded however that boxplots of Figure 5 excludes the models that did not fit at all the datasets (Figure 1) and is therefore meaningful for all models except Yamada exponential, Gompertz, and Generalized Goel. Musa Okumoto remains the best performer. Next to it the other models have also narrow boxplot (always better than 0.9, the threshold depicted as a colour horizontal line) but some outliers. It should be noted that all models behave extremely well on the OSS datasets except Generalized Goel.

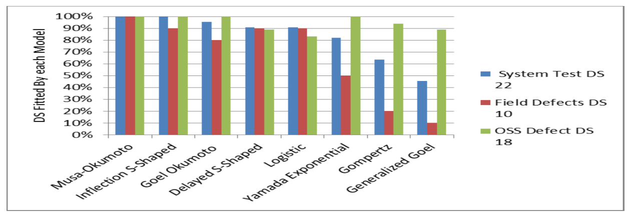

Figure 1. **No of DS fitted by each Model**

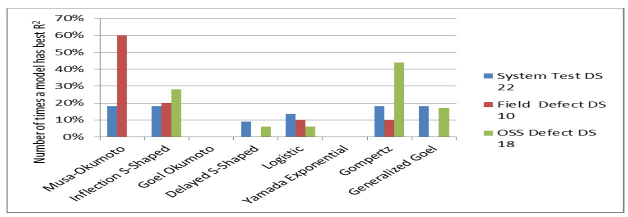

Figure 2. **Ranking on Best Fitting-R^2**

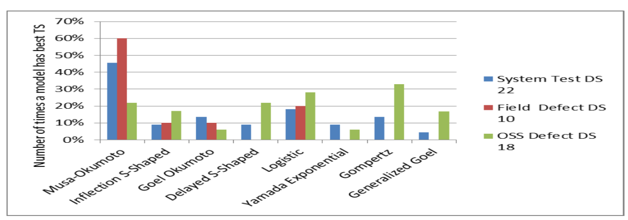

Figure 3. **Ranking on best prediction: TS**

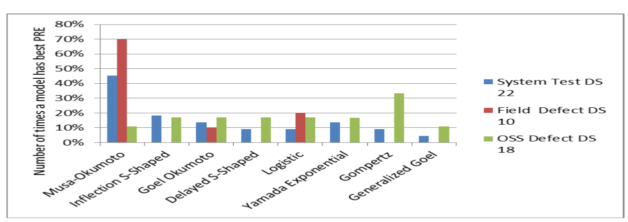

Figure 4. **Ranking on best prediction: PRE**

B. RQ2. Which SRGM models are good predictors?

For models that could fit the data set we used the first two-third data points of the data sets to train the model, and predicted the last third. We analyze their predicting capability in terms of accuracy and precision.

Accuracy:

Figure 3 shows the number of times a model is the best predictor in terms of TS. It is clear that Musa outperforms the others in both the industrial datasets. The Logistic is directly behind it. On the contrary in the case of OSS we got several ties (this explains why the sum of percentage might be greater than 100%), and the best model is Gompertz directly followed by Logistic. Figure 6 reports the TS values for all datasets. The red line represents the 0.1 threshold, usually considered indicator of good accuracy. Figure 6 allows us to discuss good models, instead of best model as in Figure 3.

Correctness:

Looking at ranks in the bar chart diagram in Figure 4, we observe that Musa Okumoto is the best model in terms of correctness in the majority of both system test and field defects dataset. Inflection S-Shaped is directly behind Musa Okumoto in the case of system test data sets while Logistic is directly behind this in the case of field defect data sets. On contrary in the case of OSS Gompertz is the best one. Apart from this there is no clear winner. Figure 7 reports the PRE values for all datasets. The red line represents the ±10% threshold, usually considered indicator of good correctness. Figure 7 allows us to discuss good models, instead of best model as in Figure 4.

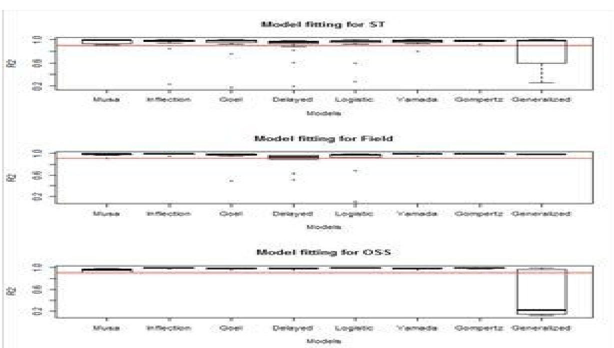

Figure 5. **Box Plots of fitting (R^2) values**

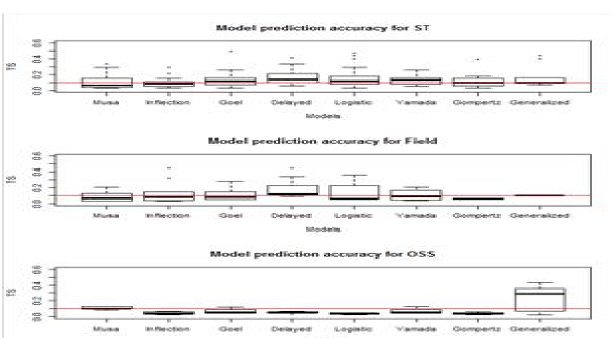

Figure 6. **Box Plots of Prediction Accuracy (TS) values**

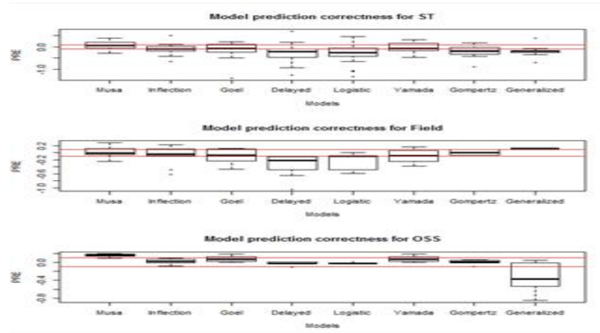

Figure 7. **Box Plots of Prediction Correctness (PRE) values**

C. RQ3. A model with good fit is also a good predictor?

Here we fit models on two thirds of the data sets and we analyse if the ones with good R^2 are also good predictors. Table 2 reports models and data sets. A model is described by three cells per category of dataset. The first contains the number of times a model fits the dataset, with any R^2 (information is also in Figure 1), the second shows how many times the model fits a data set with R^2 better than 0.9 and predicts with TS < 0.1, the third shows how many times the model fits a data set with R^2 better than 0.9 and predicts with PRE within 10%. . We observe from Table 2 that:

• On industrial datasets, Musa and Inflection are the ones with better prediction capability – however this happens only in a bit more than half the datasets

• On OSS datasets Gompertz and Inflection have good prediction capability for all datasets, followed by Logistic and Goel Okumoto.

Table II
FITTING AND PREDICTION CAPABILITY OF MODELS

	System Test DS			Field DS			OSS DS		
Model	Fitted DS	$R^2 \geq$ 0.9 AND TS < 0.1	$R^2 \geq$ 0.9 AND PRE within +/- 0.1	Fitted DS	$R^2 \geq$ 0.9 AND TS < 0.1	$R^2 \geq$ 0.9 AND PRE within +/- 0.1	Fitted DS	$R^2 \geq$ 0.9 AND TS < 0.1	$R^2 \geq$ 0.9 AND PRE within +/- 0.1
Musa	22/22	13/22	12/22	10/10	7/10	5/10	18/18	4/18	1/18
Inflection	22/22	15/22	10/22	9/10	6/10	4/10	18/18	7/18	7/18
Geol	21/22	7/22	6/22	8/10	6/10	3/10	18/18	5/18	6/18
Delayed	20/22	6/22	3/22	9/10	1/10	2/10	16/18	8/18	1/18
Logistic	20/22	7/22	5/22	9/10	5/10	5/10	15/18	5/18	5/18
Yamada	18/22	3/22	4/22	5/10	2/10	1/10	18/18	5/18	6/18
Gompertz	14/22	6/22	3/22	2/10	2/10	2/10	17/18	8/18	8/18
Generalized	10/22	2/22	1/22	1/10	1/10	0/10	16/18	4/18	2/18

VI. CONCLUSION

We have studied selected SRGM in generalized way for the purpose to derive general conclusion. In the literature nobody has validated the SRM in such generalized way: at maximum four/five models are validated on two/three datasets [28, 24, 10]. We found that the Musa-Okumoto model is the best one in fitting and predictions in industrial datasets.

Although also Inflection S-Shaped achieved very good results with respect to the metrics thresholds we adopted. The Musa-Okumoto model did not hold the same performances in OSS data, in which the Gompertz model applied better, followed by Inflection S-Shaped. We also observed two other interesting facts: 1) models which have good performances with system test data sets also good performances with field defect data, and 2) models that fit very well system test data not always predict with same performances. The practical consequences and recommendations to quality and maintenance managers are: 1) choose only one model regardless of the phase of the software lifecycle, 2) identify and choose a model that is flexible enough in case the quality process is under definition or in generable susceptible of important changes. Our future work will be devoted to the extension of the datasets used to increase the generalizability of these findings, especially in OSS projects where results and structure of datasets were very different from industrial ones.

REFERENCES

[1] 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.

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

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

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

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

[6] 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.

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

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

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

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

[11] Kapil Sharma, et al, "Selection of Optimal Software Reliability Growth Models Using a Distance Based Approach", IEEE TRANSACTIONS ON RELIABILITY, VOL. 59, NO. 2, JUNE 2010.

[12] J. D. Musa, A. Iannino, and K. Okumoto, Software Reliability Measurement, Prediction, Application. : McGraw Hill, 1987.

[13] G. H. Schick and R. W. Wolverton, "An analysis of competing software reliability models," IEEE Trans. Software Engineering, pp. 104–120, March 1978.

[14] A. N. Sukert, "Empirical validation of three software errors predictions models," IEEE Transaction on Reliability, pp. 199–205, 1979.

[15] S. Brocklehurst, P. Y. Chan, B. Littlewood, and J. Snell, "Recalibrating software reliability models," IEEE Trans. Softw. Engineering, vol. SE-16, no. 4, pp. 458–470, April 1990.

[16] A. L Goel, "Software reliability models: assumption, limitations, and applicability," IEEE Transaction on Software Engineering, pp. 1411–1423, December 1985.

[17] Abdel-Ghaly, P.Y. Chan, and B. Littlewood, "Evaluation of competing software reliability predictions," IEEE Trans. Softw. Engineering, vol. SE-12, no. 12, pp. 950–967, September 1986.

[18] T. M. Khoshgoftaar, "On model selection in software reliability," in 8th Symposium in Computational Statistics, August 1988, pp. 13–14.

[19] T. M. Khoshgoftaar and T. Woodcock, "Software reliability model selection: A case study," in Proc. of the 2nd International Symposium on Software Reliability Engineering. Austin, TX: IEEE Computer Society Press, 1991, pp. 183–191.

[20] K. Pillai and V. S. S. Nair, "A model for software development effort and cost estimation," IEEE Transaction on Software Engineering, vol. 23, no. 8, pp. 485–497, 1997.

[21] 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.

[22] 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.

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

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

[25] 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.

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

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

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

[29] Basili, et al, "The Goal Question Metric Approach", in Encyclopedia of Software Engineering, Wiley, 1994.

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

[31] 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.

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