Software Defect Prediction using Maximum Likelihood Estimator and Non-linear Regression by Analyzing Software Reliability Growth Model.

Meer Tauseef Ali[1] and Dr .Syed Abdul Sattar[2]
1. Research Scholar,(PP.COMP.SCI.O398) Department of Computer Science, Rayalaseema University, Kurnool, A.P.
2. Professor & Principal, Nawab Shah Alam Khan College of Engg. & Tech., Hyderabad

Abstract—SRGMs can be used during the project to help make testing resource allocation decisions and/ or it can be used after the testing phase to determine the latent faults prediction to assess the maturity of software artifact. Software Reliability Growth Models (SRGMs) have been used by engineers and managers for tracking and managing the reliability change of software to ensure required standard of quality is achieved before the software is released to the customer. Two of the widely known and recommended techniques for parameter estimation are maximum likelihood and method of least squares. In this paper we compare between the two estimation procedures for their usability and applicability in context of SRGMs. We also highlight a couple of practical considerations, reliability practitioners must be aware of when applying SRGMs

Keywords: Predictive relative error(PRE), unbiased, BPRE, Non-linear Regression, Maximum likelihood estimation.

1.Introduction

Software is playing an ever increasing role in our day today life. Most of the products and services we consume are now based on software or uses software in certain ways [1].

1.1Software Defect Prediction Techniques

Software Defect Prediction (SDP) techniques are used either to classify which modules are defect-prone or to predict the number of defects expected to be found in a software module/project. For classification of defects or predicting defects, various numbers of techniques have been used in software and they are broadly classified into two groups. One used for predicting expected number of defects to be found in a given software artifact (Prediction) and techniques that are used to predict if or not a given software artifact is likely to contain a defect (Classification). Figure 1 illustrates commonly used software defect prediction techniques grouped according to the purpose – defect count prediction or defect prone classification.

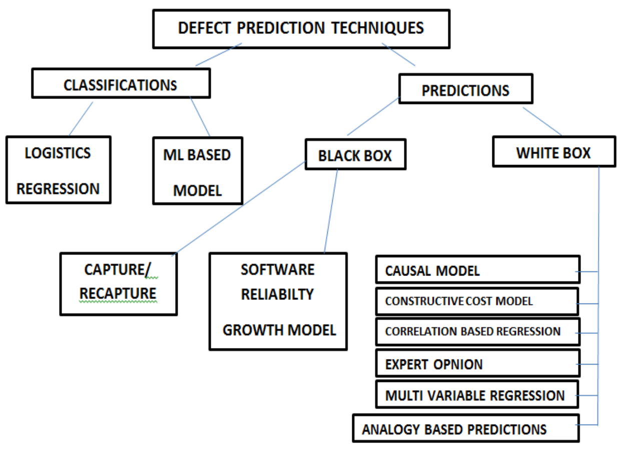

***Figure 1**: Overview of different software defect prediction techniques*

There is a huge demand of software system, dependability on software increases many fold thereby there is a huge complexity as software advances. The link between complexity and software faults have been suggested for long, studies as early as 1980s such as Software complexity and its impact on software reliability by K. S. Lew, T. S. Dillon, and K. E. Forward [2] suggest that software complexity often affects its reliability. Thus while it is important to keep the complexity of software under check, it is also important to tack and monitor their reliability growth.

Software testing is still the main source of ensuring reliability and quality of software systems. As per M. Camuffo, M. Maiocchi, and M. Morselli testing in the area of software products is highly resource intensive exercise; some of the estimates put it around 50% of overall development cost [3]. But, however according to C.T. Lin and C.Y. Huang testing resource consumptions can be much more resource/cost efficient, if project managers are able to plan testing activities well [4]. Since, According to T. Goradia the software can rarely be made fully error free, project managers need to balance costs associated with software testing to cost of fixing bugs after release [5]. To eliminate the reliability change in software products and to use the reliability growth prediction for making testing allocation decision, SRGM have been used.

While applying SRGM to defect lot of difficulties may be encountered, while using two types of estimates namely NLR &MLE, There is a need for practical explanation. We compare between the two and introduce a measure for assessing the predictive power of reliability models. The data used for this study is time-domain failure data for a real-time control system provided in [6] and used in many earlier studies including [7],[8]. In the data 136 faults have been reported with their time between failures (TBF).

2. Background

2.1 SRGMs: Software Reliability Growth Models

Software reliability engineering tends to focus on using engineering techniques for assessing and improving the reliability of software systems during development and post development. A roadmap on the software reliability engineering is presented in [9]. SRGMs) or the dynamic models generally use statistical distributions of the defect inflow patterns to estimate/predict the end-product reliability [10]. The SRMs and SRGMs could also be differentiated based on their access to source code which former being a white box models while the latter being black box modeling of software reliability. We focus on SRGMs in this study.

2.2 Model Selection

Since the start of reliability modeling within software domain in early 1960s [11], a number of SRGMs have been proposed and evaluated [5]. For assuming & improving the reliability of the software system during development process and post development, software reliability estimates tends to focus in engineering techniques. Studies such as by Goel [12] and Musa [13] have shown that different models/families of models are better suited than others for certain applications.

2.3 Comparing between SRGMs

By adapting a comparative study only, We can understand the different models and their ability to find & predicts given defect data. A number of NHPP based SRGMs have been reviewed and compared on their fit and predictive power by Pham [7]. Ullah et al. [14] also present a study comparing eight SRGMs onto large dataset consisting of fifty defect data from industrial and open source projects.

Other studies have also evaluated and compared different SRGMs on industrial data, Although a number of studies have compared and evaluated different SRGMs within different context, we are still far from making a consensus on how to select SRGMs for given purpose and which models are best for given process characteristics. The situation with different SRGMs comparison is very well summarized by Stephan Kan as: "Some models sometimes give good results, some are almost universally awful, and none can be trusted to be accurate at all times." [10].

2.4 Parameter Estimation

Two practical and important challenges faced when applying SRGMs in practice/industry are the process to be followed and how to estimate the parameters. IEEE standard 1633: recommended practice on software reliability [14] provides a 13-steps procedure on assessing and predicting the software reliability. The standard also lists three methods commonly used for parameter estimation when using SRGMs as: method of moments, least squares and the maximum likelihood estimation. Maximum likelihood estimation is the recommended approach by the standard and by the various studies introducing new SRGMs [12], [16], [17].Parameter estimation using Maximum likelihood estimation requires solving sets of simultaneous equations to maximize the likelihood of defect data coming from given function (model equation) to find the parameters. [18].

The problem of using MLE widely for parameter estimation is further compounded either due to SRGM models with complex log-likelihood functions and cases where MLE does not converge to give unique estimation of unknown parameters. Meyfroyt [19] provides necessary and sufficient conditions for ensuring unique, positive and finite estimation of parameters using MLE for Goel-Okumoto, Yamada S-shaped and Inflection S-shaped models. While applying SRGM to defect lot of difficulties may be encountered, while using two types of estimates namely NLR &MLE, There is a need for practical explanation

On the other hand the least square estimation uses curve fitting to the observed data for making estimation of unknown parameters. Parameters values are estimated for curve that gives minimum sum of square of errors, i.e. curve that fits best (with respect to sum of squared errors). Given the nature of common SRGMs the least square estimation usually leads to using non-linear regression (NLR) for estimating the unknown parameters. Contrary to MLE, least square estimation is easy to apply, and NLR is often available as standard routine in most commercially available statistical packages.

To eliminate the reliability change in software products and to use the reliability growth prediction for making testing allocation decision SRGM have been used. It can be safely assumed that statistically MLE is much better parameter prediction procedure than least square, but the least square is much easier and provide consistent results in wider data sets and thus a preferred method of choice by industrial practitioners. Also in certain cases where MLE cannot provide the parameter estimations, least square approach is the natural alternative. Thus the least square estimator/NLR is also used more often than MLE for studies evaluating different SRGMs over large datasets [14], Given the differences between the two estimators the need to understand the applicability and performance differences of these two estimators is quite apparent.

3. Research Context and Method

3.1 Research Objectives

In this study we take a look at some of the practical considerations and questions faced by software reliability practitioners. The objective is mainly to document these aspects and mark their importance. Mainly by comparing the MLE & NLR as this procedure provide estimation of unknown SGRM model parameters. but further assessing the predictive relative error metric. We also comment on reproducibility of earlier studies from literature and provide directions for further research.

3.2 SRGMs and Data

In this study we use three of the very early and widely used software reliability models, the SRGMs used and their mean value functions are listed below in Table 1.

The main reason for their selection is their wide familiarity and availability of MLE simultaneous equations. The mean value functions have parameters a, which refers to total number of predicted defects and b, which is generally the shape parameter or growth rate parameter. Parameter β in Inflection S-shaped model is assumed to be 1.2 following the earlier studies [7].

Table 1: Summary of SRGMs used in this study

S.No	Model Name	Mean Value Function	Ref
1	Goel-Okumoto (GO)	$m(t)= a\ (1-e^{-bt})$	16
2	Delayed S-shaped model	$m(t)= a\ (1-(1+bt\ )e^{-bt})$	17
3	Inflection S-shaped model	$m(t)= a\ (1-e^{-bt}\ )/\ (1+\beta e^{-bt}\ )$	12

The data used for this study is time-domain failure data for a real-time control system provided in [6] and used in many studies including [7][8]. In the data 136 faults have been reported with their time between failures (TBF). For practical reasons we also assume 136 to be the real asymptote of given data, i.e. actual total number of defects. Cumulative time obtained by successively adding TBF is used for fitting the cumulative distribution functions to different SRGMs. 122 failures are used for fitting the data and making parameter estimates, while the rest are used to evaluate the predictive power.

3.3 Data Analysis Techniques

To ensure high reproducibility we list all the data analysis techniques and equations used for analysis in this study with their references.

1. For parameter estimation using least squares we use Non-Linear Regression routine available in statistical package IBM SPSS, the starting values provided were and iterations were done until successive residual errors difference was less than (default value in SPSS).

2. For parameter estimation using MLE, we use package maxLik, a package for statistical environment R[20]. The optimization method used was Nelder-Mead (NM) and the starting values provided were same as those used for NLR routine.

3. We also compare the parameter estimations obtained by above methods (NLR and MLE using maxLik) with earlier study by Pham [7] using the same data.

4. To make the two estimators comparison even more robust, we further use the non-linear simultaneous equations for getting the analytical solution using MLE. The equations are available for Goel-Okumoto model and Delayed S-shaped model described in [21] and reproduced below:-For GO model:

$$(\ n\ /a\ )=\ (1-e\ )^{-bs}{}_{n} \quad \ldots\ldots\ldots\ldots\ldots(1)$$

$$(n\ /\ b\ )=\ \sum_{1}^{n} s_{\,i} + a\ s_{\,n}\ e^{-bs}{}_{n}\ \ldots\ldots\ldots\ldots(2)$$

For Delayed S-shaped model

$$(\ n\ /a\ )=\ 1-(1+bs_n\ )e^{-bs}{}_{n}\ \ldots\ldots\ldots(3)$$

$$(2n\ /\ b\ )=\ \sum_{1}^{n} s_{\,i} + abs_n{}^{2\cdot}e^{-bs}{}_{n}\ \ldots\ldots(4)$$

Where n represents number of failures reported; time between failures is represented as (t-k; k=1,2,……n) and where time to kth failure is given by $s= \sum_{1}^{n}\ t$; for details refer to [21].

Equations (1) & (2) or (3) & (4) can be solved simultaneously to obtain the point estimates of parameters. We used Matlab fsolve to solve system of non-linear equations given above.

5. To make comparison of asymptote prediction accuracy, we use the metric Predicted Relative Error (PRE), which is described in the IEEE standard 1633 and also used in earlier studies as measure of prediction accuracy [14].PRE is defined as ratio between predicted error (predicted minus the actual asymptote) to the predicted number of failures

PRE=((predicted–actual)/predicted) ….(5)

PRE resolves a common problem with using relative error for comparing between different models prediction, the relative error is the ratio of prediction error over actual value and thus if the predicted value is much larger (in multiples) than the actual value, relative error can be greater than 100%. PRE provides a comparative scale between [-1 1] or [-100% 100%], where value close to zero means better predictive accuracy and closer to +/-100% is as worse prediction as it can get.

Although we identify one major problem with PRE, which is: It provides asymmetric value based on over or under prediction. The problem can be easily understood using a simple example.

Let us assume actual value be and case1: the predicted value is 20% higher than actual i.e. 1.2a; for case2: the predicted value is 20% lower than actual (i.e. -20% of actual or 0.8a).

Now applying PRE to case1 and case2, gives PRE values:-

PRE(case1)=((1,2aa)/1.2a)=(0.2/1.2)=0.166667 or 16.67%

While for

PRE(case2)= ((0,8a–a) / 0.8a) =(- 0.2 /0.8)=-0.25 or -25%

To make PRE symmetric and thus give consistent value for over and under estimation we define BPRE, referring to Balanced Predicted Relative Error, as follows (equation (6)):

BPRE =((Predicted -Actual)/((n*Predicted+(1n)(2*Actual-Predicted)))

Where n { 1 if Predicted >Actual or 0 if if Predicted < Actual ……..(6).

Now applying above defined BPRE to same case1 and case2 , gives BPRE values:-

PRE(case1)= ((0,8a–a) / 0.8a) =(- 0.2 /0.8)=-0.25 or -25% ,

and

PRE(case2)= ((0,8a–a) / 0.8a) =(- 0.2 /0.8)=-0.25 or -25%

If we look at the method to estimate parameters value in software predictions model provided by Miyazaki et al { 23}, Defined Ri a balanced relative (BRE) metric which is also referred as BRE bias has defined as given by

$\mathbf{R_i}$ or $BPRE_{bias}$ = (Actual - Predicted) /((min(Actual ,Predicted)) ……….(7).

Our metric BPRE is similar to $\mathbf{R_i}$, but different in the sense that while $\mathbf{R_i}$ is unbounded on both sides, BPRE is bounded [-0.5, 1) which is useful to make comparisons when deviations are particularly large compared to actual values.

6. To compare the model fitting to data for both fit and predicted values, we use another widely used metric, Mean Square Error (MSE). Mean square error measures the average deviations between the predicted and actual values [23], thus a measure of fit, it is given by equation (8):

$$MSE=\left(\sum_{1}^{n} (a_i-p_i) /(k-q)\right) \quad ………(8).$$

Where a_i is actual values, p_i predicted values for data set of size k and q is the number of parameters. Where a_i is actual values, p_i predicted values for data set of size and is the number of parameters.

4. Results

4.1 Parameter estimation using MLE and NLR. estimation

Table 2: Comparing Parameters With Different Estimators

Asymptote	**MLE**	**NLR**	**Pham**	**Equation**
Goel-Okumoto	132	114.05	125	139.37
Delayed S	132	103.33	140	125.16
Inflection S	132	107.6	135.5	
Growth Rate	**MLE**	**NLR**	**Pham**	**Equation**
Goel-Okumoto	3.80E-05	6.07E-05	6.00E-05	3.65E-05
Delayed S	9.73E-05	1.66E-05	7.00E-05	9.76E-05
Inflection S	5.79E-05	1.07E-05	7.00E-05	

Parameter estimation using maximum likelihood and non-liner regression procedure are summarized in Table 2. The table also provides comparison of parameters values obtained in study using same data by Pham [7] and also by solving MLE simultaneous equations provided in [23].

Form Table 2 it can be observed that the asymptote, total number of predicted defects/failures) predictions obtained in this study using maximum likelihood estimator utilizing package maxLik gives very consistent results for all three models. While the asymptote predictions using non–linear regression routine (NLR) varies much more with minimum prediction being 103 for Delayed S-shaped model and 114 for GO model.

It is further interesting to note that significant differences are also observed between our predictions using (MLE) and values obtained by earlier study by Pham, although in both case the estimator used is the same (MLE). The difference observed here may be attributed to difference in tools used or the starting values predicted. Given that the tool used and starting values details are not available for earlier study, it is difficult to verify the source of this observed difference.

Predictions for growth rate parameter ( ) with different estimators are also listed in Table 8. While there are variations between different models growth rates obtained in this study using MLE and NLR.

The growth rate is predicted to have highest value for Delayed S-shaped model and lowest for GO model using both (MLE & NLR) estimators in our study. The growth rates predicted in Pham study are closer to each other. It can also be noted that for both asymptote and growth rate, our estimates using MLE are very close to the parameters estimates obtained using MLE simultaneous equations described earlier.

The fitting of predicted models using different estimators to actual data is also represented in Fig 1, Fig 2 and Fig 3.

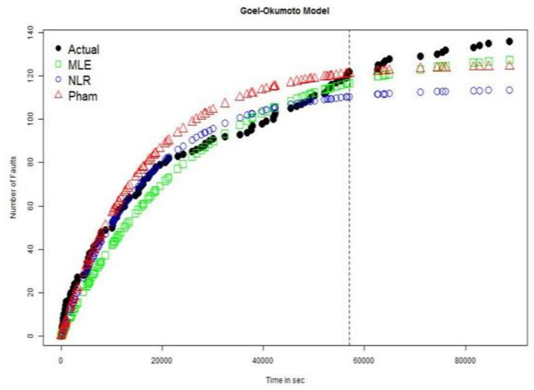

Figure 1: Goel-Okumoto model fitting to data with different estimators

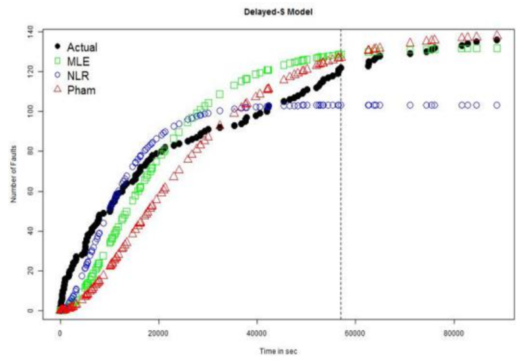

Figure 2: Delayed S-shaped model fitting to data with different estimators

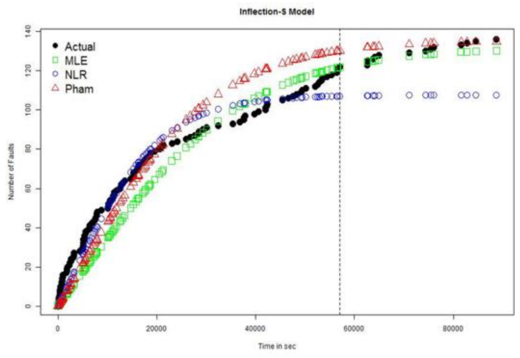

Figure 3: Inflection S-shaped model fitting to data with different estimators

4.2 Predictive Accuracy using Predicted Relative Error (PRE) and unbiased PRE (BPRE)

It is interesting to note from Table 3 that all but one estimate under predicts for given dataset. Using PRE and BPRE values for same parameter predictions we can also see that BPRE gives better and more accurate representation of undervalued asymptote prediction as described in the section 3. The BPRE values for asymptote predictions using MLE and NLR are also presented below in Figure 5. We also add two more models using NLR procedure to make further check.

We now compare the predictive accuracy of asymptote values obtained using MLE estimator to NLR estimators.

Table 3: PRE and BPRE for different estimators and models

Asymptote	PREMLE	NLR	Pham
Goel-Okumoto	-3.00%	-19.20%	-8.80%
Delayed S	-3.00%	-31.60%	2.90%
Inflection S	-3.00%	-26.40%	-0.40%
Asymptote	**BPRE**	**NLR**	**Pham**
Goel-Okumoto	-2.90%	-13.90%	-7.50%
Delayed S	-2.90%	-19.40%	2.90%
Inflection S	-2.90%	-17.30%	-0.40%

Figure 5 shows that in our study although MLE estimators also under predict asymptote values the prediction is consistent for all models and prediction accuracy much higher (BPRE lower than -5%).

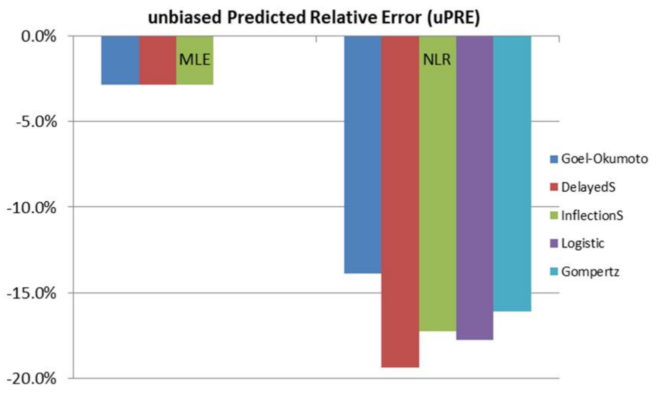

Figure 5: Comparing between BPRE values for MLE and NLR estimations

While the unbiased predictive relative error value for NLR estimators is comparatively higher closer to but under negative 20% for different models tested here.

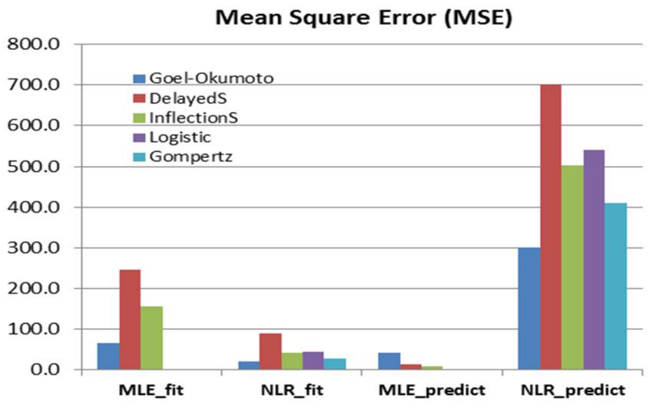

Figure 6: Comparing between MSE fit and predict for MLE and NLR estimation

From Table 4 and Figure 6 we can observe that MSE fit values using NLR are much better compared to values obtained using MLE. This is not surprising given that least square procedure actually minimizes the sum of square of errors between the observed data and used model. On comparing MSE values using MLE obtained in this study to earlier study by Pham and by using equations, we can see that in all but one case MSE values obtained in this study are much smaller than those presented in earlier study and they are also closer to values obtained using MLE simultaneous equations.

Further the interesting point to note from the comparison is that despite NLR giving very good fit values, it does comparatively worse for the MSE values for the predicted values. Mean square error using MLE are significantly smaller to ones obtained using NLR which confirms that MLE is a better estimator for prediction purposes.

Although as described earlier that SSE (Sum of Squared Errors)/MSE is not a fair comparison parameter between MLE and NLR for the fitted data points, but since

MSE for the predicted data is not optimized for both estimators (MLE & NLR), it serves the purpose of comparing between the two estimators on evaluating fit of given model to observed data and goodness-of-fit to predicted data.

4.3 Which Estimators give better Fit to data and Predicted values

Table 4: Comparing MSE fit and predict values for different estimators and models

MSE fit	MLE	NLR	Pham	Equation
Goel-Okumoto	67	20.8	62.7	65.4
Delayed S	246.6	89.2	420.4	223.8
Inflection S	155.7	42.3	132.1	
MSE predict	**MLE**	**NLR**	**Pham**	**Equation**
Goel-Okumoto	42.7	301.6	50.4	1.6
Delayed S	12.8	702.0	22.5	40.9
Inflection S	9.3	501.6	23.0	

Another widely used parameter to compare different models and their estimators for their performance is their ability to fit the observed defect/failure data and to the predicted the data. Mean Square Error (MSE) is often used to compare the fit of observed and predicted values. MSE and values obtained for MLE and NLR estimators are provided in Table 4. The MSE values using MLE and NLR estimation using additional Logistic and Gompertz model (for NLR estimator) is also presented in Figure 6.

5. Conclusions

It is noted in the study that while MLE is the recommended estimator with superior statistical properties, its usability and applicability in all situations is questionable. Further MLE is difficult to apply which limits its use in industry, especially due to lack of tools support. In this study using data from literature we have compared between two of the most widely recommended and used methodology for estimating parameters for the purpose of applying SRGMs to defect/failure data. The study provides useful and practical insights for industry practitioners and early researchers applying reliability modeling to defect/failure data.

With results in this study suggesting that the fit, predict and predictive accuracy obtained using MLE and NLR estimators may be much different from one estimator to another, more research in this direction is needed to establish these differences in different contexts and thus helping to resolve the dilemma faced by reliability practitioners of which estimator to use and in which conditions a given estimator is better than other.

Initial results presented here and properties of MLE and NLR estimators suggest that while NLR is good estimator for fitting the data to observed failure data, MLE is better estimator for making reliable predictions.

References:

[1]. R.Kitchin and M. Dodge, Code/space: Software and everyday life. The MIT Press, 2011.
[2]. K.S.Lew,T. S. Dillon,and K.E.Forward, “Software complexity and its impact on software reliability,” IEEE Transactions on Software Engineering, vol. 14, no. 11, pp. 1645–1655, 1988.
[3]. M.Camuffo,M.Maiocchi,andMorselli,“Automatic software test generation,” Information and Software Technology, vol. 32, no. 5, pp. 337– 346, 1990.
[4]. C.T.Lin and C.Y.Huang,“Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models,” Journal of Systems and Software, vol. 81, no. 6, pp. 1025–1038, 2008.
[5]. T.Goradia,“Dynamic impact analysis: A cost-effective technique to enforce error-propagation,” ACM SIGSOFT Software EngineeringNotes,vol.18,no.3,pp.171–181,1993.
[6]. M.R. Lyu ,Handbook of software reliability engineering, vol. 3. IEEE Computer Society Press CA, 1996.
[7]. H.Pham,“Software reliability and cost models: Perspectives, comparison, and practice,” European Journal of Operational Research, vol.149,no.3,pp.475–489,2003.
[8]. X.Zhang,X.Teng,and .Pham,“Considering fault removal efficiency in software reliability assessment,” IEEE Transactions on Systems, Man and Cybernetics, Part A:Systems and Humans,vol.33,no.1,pp.114–120,2003.
[9]. M.R.Lyu,“Softwarereliability engineering: A roadmap,” in Future of Software Engineering, 2007. FOSE’07, 2007, pp. 153–170.
[10]. S.H.Kan and others,Metrics and Models in Software Quality Engineering, 2/e. Pearson Education India, 2003.
[11]. Kapur,H.Pham,S.Anand,and.Yadav,“A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation,”IEEE Transactions on Reliability,vol.60,no.1,pp.331–340,2011.
[12]. A.L.Goel,“Software reliability models: Assumptions, limitations, and applicability,” IEEE Transactions on Software Engineering, no.12,pp.1411–1423,1985.
[13]. J.D.Musa, A.Iannino, and K.Okumoto, Software reliability. McGraw-Hill New York, 1987.
[14]. N.Ullah, M.Morisio, and A.Vetro, “A Comparative Analysis of Software Reliability Growth Models using Defects Data of Closed and Open Source Software,”in 35th Annual IEEE Software Engineering Workshop (SEW), 2012, pp. 187–192.
[15]. IEEE Reliability Society, “IEEE Recommended Practice on Software Reliability.” The Institute of Electrical and Electronics Engineers, Inc,2008.
[16]. A.L.Goel and K. Okumoto,“Time-dependent error-detection rate model for software reliability and other performance measures,” IEEE Transactions on Reliability,vol. 28,no.3,pp.206–211, 1979.
[17]. S.Yamada,M.Ohba, and S.Osaki,“S-shaped reliability growth modeling for software error detection,” IEEE Transactions on Reliability,vol.32,no.5,pp.475–484, 1983.
[18]. D.C. Boes, F. A. Graybill, and A. M. Mood, “Introduction to the Theory of Statistics,” Series in probabili, 1974.
[19]. P.H. A. Meyfroyt, “Parameter Estimation for Software Reliability Models,” 2012.
[20]. A.Henningsen and O. Toomet, “maxLik: A package for maximum likelihood estimation in R,” Computational Statistics, vol. 26, no. 3, pp. 443–458, 2011.
[21]. S.S. Gokhale and K. S. Trivedi, “Log-logistic software reliability growth model,” in Proceedings of Third IEEE International High-Assurance Systems Engineering Symposium, 1998., 1998, pp. 34–41.
[22]. Y.Miyazaki, A. Takanou, H. Nozaki, N. Nakagawa, and K. Okada, “Method to estimate parameter values in software prediction models,” Information and Software Technology, vol. 33, no. 3, pp. 239–243, Apr. 1991.
[23]. S. Hwang and H. Pham, “Quasi-renewal time-delay fault-removal consideration in software reliability modeling,” IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans., vol. 39, no. 1, pp. 200– 209, 2009.