IJCSC 0973-7391 Volume 6 • Number 1 Sep - Mar 2015 pp.73-78 available online at www.csjournalss.com

Study of Perfect and Imperfect Debugging NHPP SRGMs used for Prediction of Faults in a Software

Ravneet Kaur[1], Poonam Panwar[2]
[1]M.Tech Student, [2]Assistant Professor,
Deptt. of Comp. Sc. & Engg, Ambala College of Engg and Applied Research, Ambala, India.
rdhillon453@gmail.com, rana.poonam1@gmail.com

Abstract: Various NHPP SRGMs have been studied with various assumptions. Many of the SRGMs assume that each time a failure occurs, the fault that caused it is immediately removed and no new faults are introduced called perfect debugging. But in reality, it is possible that the fault which is supposed to have been removed may cause a failure or new faults may get introduced in software known as imperfect debugging. In this paper we have studied various perfect and imperfect debugging models. An introduction of parameter estimation techniques and goodness of fit parameter is also discussed in our proposed study.
Keywords: Imperfect debugging, parameter estimation, goodness of fit.

I. INTRODUCTION

Software is a key part of many critical applications. Software quality is a major concern in software development process. The quality of software means conformance to requirements, fitness for the purpose and level of satisfaction. The issue of estimating and improving software quality has become the primary concern for both software developers and product users. For software developers, releasing reliable software products into the market on time and within budget is a must to secure financial growth and to gain competitive edge in the industry. For software users, failure-free operation of the software guarantees avoidance of inconvenience and loss of valuable time. This leads to the fact that software reliability is now an important research area, and it is imperative to have sound methodologies to measure, quantify, and improve software quality. According to the ANSI definition Software reliability is the probability of failure-free software operation for a specified period of time in a specified environment [1] [2].

Measuring and computing software reliability can be used for planning and controlling all testing resources during software development. To assess software reliability and assure software quality one of the current methods is to apply SRGMs. SRGMs are the one particular aspect of Software Reliability Engineering (SRE) that has received the most attention. SRGMs can provide quantitative information about how to improve the reliability of software products, and greatly help software engineers to measure the defect levels, failure rates and reliability during the coding and testing phases. Furthermore, SRGMs can help project managers to determine the testing resources and manpower needed to achieve desired reliability requirements [3] [4]. An important class of SRGMs that has been widely studied is Non-Homogenous Poisson Process (NHPP). It forms one of the main classes of existing SRGMs due to its mathematical tractability and wide applicability. NHPP models are useful in describing failure processes, providing trends such as reliability growth and fault-content. NHPP models determine an appropriate mean value function to denote the expected number of failures experienced up to a certain time. NHPP SRGMs are used extensively for reliability estimation of software. Various NHPP SRGMs have been studied with various assumptions. Many of the SRGMs assume that each time a failure occurs, the fault that caused it can be immediately removed and no new faults are introduced which is known as perfect debugging. But in reality, it is possible that the fault which is supposed to have been removed may cause a new failure. In this paper, we consider that whenever a failure occurs the detected fault is not perfectly removed and there is a chance of raising new fault due to wrong diagnosis or incorrect modifications in the software which is called imperfect debugging.

The paper gives general idea of perfect debugging, imperfect debugging and their effect on fault content of software. This also focuses on parameter estimation techniques and goodness-of-fit criteria used for model evaluation. The remainder of this paper is organized as follows. In Section II we discussed the perfect and imperfect debugging SRGMs. Section III deals with parameter estimation techniques and goodness-of-fit criteria used in SRGMs. Finally, Section IV concludes the paper.

II. DEBUGGING

Debugging is the process of fault detection and correction. There are two type of debugging process (i) Perfect and (ii) Imperfect Debugging.

Perfect Debugging

In perfect debugging it is considered that faults are corrected with certainties which are responsible for software failures and no new faults are introduced. Most of the earlier software reliability models assume the fault removal process to be perfect i.e. when an attempt is made to remove a fault, it is removed with certainty and no new faults are introduced.

In one of the earliest papers of software reliability by Jelinski and Moranda (1972), software failure rate is assumed to be proportional to the residual number of bugs, and each bug has a constant failure rate contribution. In addition, the number of bugs decreases by one after each software failure to indicate a perfect removal of the bug that caused software failure [5]. Some of the perfect debugging NHPP SRGMs are:

Yamada Delayed S-Shaped Model [6]: The model is based on NHPP with a different mean value function to reflect the delay in failure reporting. Its mean value function is S-shaped and is given below:

$$m(t) = a(1 - (1 + bt)e^{-bt}$$

Goel-Okumoto Model: This model was proposed by Goel-Okumoto [6] they consider failure detection as a non homogeneous Poisson process with an exponentially decaying rate function. The mean value function is given as:

$$m(t) = a(1 - e^{-bt})$$

Generalized Goel NHPP Model [6]: In this model exponential distribution assumes a pattern of decreasing defect rates or failures cases in which the failure rate first increases and then decreases. Goel proposed a generalization of Goel-Okumoto model with an additional parameter c. Its mean value function is given as:

$$m(t) = a(1 - e^{-bt^c})$$

Gompertz growth Curve Model: Gompertz model [7] is used in the Fujitsu and Numazu work. It is one of the simplest S-shaped software reliability growth models. Its mean value function is:

$$m(t) = a(k^{b^t})$$

Modified Dune Model: Duane published a report that presented failure data of several systems during their developments in 1962 by analyzing the data. It was observed that the cumulative Mean Time Between Failures versus the cumulative operating time becomes close to a straight line if plotted on log-log paper. Later, a modified Duane model [8] was proposed and its hypothesized mean value function is:

$$m(t) = a[1 - (\frac{b}{b + t})^c]$$

Inflection S-Shaped Model: This model [9] was proposed by Ohba and its underlying concept is that the observed software reliability growth becomes S-shaped if faults in a program are mutually dependent, i.e., some faults are not detectable before some others are removed. The mean value function is:

$$m(t) = \frac{a(1 - e^{-bt})}{1 + \beta e^{-bt}}$$

Logistic Growth Curve Model: In general, software reliability tends to improve and it can be treated as a growth process during the testing phase. The reliability growth occurs due to fixing faults. Therefore, under some conditions, the models developed to predict economic population growth could also be applied to predict software reliability growth. These models simply fit the cumulative number of detected faults at a given time

with a function of known form. Logistic growth model [9] is one of them and it has an S-shaped curve. Its mean value function is

$$m(t) = \frac{a}{1 + ke^{-bt}}$$

Musa-Okumoto Model: Musa-Okumoto [10] have been observed that the reduction in failure rate resulting from repair action following early failures are often greater because they tend to the most frequently occurring once and this property has been incorporated in the model. Its mean value function is given as:

$$m(t) = a \ln (1 + bt)$$

Pham Zhang IFD Model [11] mean value function is given as:

$$m(t) = a - ae^{-bt}(1 + (b + d)t + bdt^2)$$

Imperfect Debugging

Perfect debugging is an unrealistic assumption due to the human element involved in debugging a software. In addition, the complexity of certain softwares may not only allow introduction of new bugs but also the possibility that the bug(s) that caused the failure may not have been completely eliminated. Goel [12] stated it as one of the major practical limitations of existing models and modified the J-M model by introducing the concept of imperfect debugging.

The phenomenon of imperfect debugging occurs due to introduction of new faults during the correction process and also faults corrected are less than the faults responsible for failures. If some of the detected faults are not removed with certainty or new faults introduced during debugging process then it is called imperfect debugging.

There are two type of imperfect debugging possibilities-first, on a failure the corresponding fault is identified, but just because of incomplete understanding of the software, the detected fault is not removed completely and hence the fault content of the software remains unchanged on the removal action, proposed by Kapur [13] known as imperfect fault debugging, - second, when on a failure the corresponding fault is identified and removed with certainty but some new faults are added to the software during the removal process, proposed by Obha and Chou[14]. This type of imperfect debugging led to an increase in the fault content of the software known as error generation. Some commonly used imperfect debugging NHPP SRGMs are:

Pham-Nordmann-Zhang (PNZ) model [15] incorporates the imperfect debugging phenomenon by assuming that faults can be introduced during the debugging phase. This model assumes that the introduction rate is a linear function time-dependent overall fault content function and the fault detection rate function is non-decreasing time-dependent with an inflection s-shaped model. Its mean value function is given as:

$$m(t) = \frac{a(1 - e^{-bt})\left(1 - \frac{\alpha}{b}\right) + \alpha at}{1 + \beta e^{-bt}}$$

Pham-Zhang (P-Z) Model [16] assumes constant introduction rate is exponential function of testing time and the error detection function is non decreasing with an inflection S-shaped model. Its mean value function is given as:

$$m(t) = \frac{1}{(1 + \beta e^{-bt})}((c + a)(1 - e^{-bt}) - \frac{ab}{b - \alpha}(e^{-\alpha t} - e^{-bt}))$$

Yamada Exponential Model [17] attempts to account for testing effort and incorporates an exponential testing effort function. The model type is concave. Its mean value function is given as:

$$m(t) = a(1 - e^{-r\alpha(1 - e^{-\beta t})})$$

Yamada Rayleigh [17] attempts to account for testing effort and incorporates an exponential testing effort function. The model type is S-shaped. Its mean value function is given as:

$$m(t) = a(1 - e^{-r\alpha\left(1-e^{\left(-\frac{\beta t^2}{2}\right)}\right)})$$

Yamada Imperfect Debugging Model 1 [18] assumes exponential fault-content function and constant fault-detection rate. Its mean value function is given as:

$$m(t) = \frac{ab(e^{\alpha t} - e^{-bt})}{a+b}$$

Yamada Imperfect Debugging Model 2 [18] assumes constant fault-introduction rate and constant fault-detection rate. Its mean value function is given as:

$$m(t) = a(1 - e^{-bt})\left(1 - \frac{\alpha}{b}\right) + \alpha at$$

Zhang-Teng-Pham Model [19] mean value function is given as:

$$m(t) = \frac{a}{p-\beta}[(1 - \frac{(1+\alpha)e^{-bt}}{1+\alpha e^{-bt}})^{\frac{c}{b}(p-\beta)}]$$

III. PARAMETER ESTIMATION TECHNIQUES AND GOODNESS OF FIT CRITERIA USED IN SRGMs

Parameter Estimation Techniques

There are various parameter estimation techniques which used to estimate the parameters of different SRGMs. Most commonly used approaches are maximum likelihood estimation (MLE) and least square estimation (LSE). MLE is the direct parameter estimation technique which input the data directly into equations for estimating the parameters. LSE is an indirect parameter estimation technique in which fitting the curve described by the function to the data and estimating the parameters from the best fit to the curve. The Least Square Estimation [20] is used for estimating parameters by minimizing the squared discrepancies between observed data and their expected values. Unlike MLE, this method does not require distributional assumptions and useful for obtaining a descriptive measure of summarizing observed data. However it has no basis for testing hypotheses or constructing confidence intervals. The LSE method is easier to calculate by hand and easier to program. The LSE method is also traditionally associated with the use of probability plots to assess goodness-of-fit. LSE is good estimator for fitting the data to observed failure data. Because MLE generally tends to be biased but LSE can produce unbiased results so we decide to use LSE to estimate the parameters of selected models.

The Maximum Likelihood Estimation technique was originally developed by R.A. Fisher in the 1920s [21] begins with writing a mathematical expression known as the Likelihood Function of the sample data. The likelihood of a set of data is the probability of obtaining that particular set of data, given the chosen probability distribution model($L(\theta|X) = f(X|\theta)$). This expression contains the unknown model parameters. The desired probability distribution is the one that makes the observed data most likely, which means that we are interested in finding the value of the parameter vector that maximizes the likelihood function $L(\theta|X)$. When you supply distribution functions, MLE computes the parameter estimates using an iterative maximization algorithm. With some models and data, a poor choice of starting point can cause MLE to converge to a local optimum that is not the global maximizer, or to fail to converge entirely. Even in cases for which the log-likelihood is well-behaved near the global maximum, the choice of starting point is often crucial to convergence of the algorithm. In particular, if the initial parameter values are far from the MLEs, underflow in the distribution functions can lead to infinite log-likelihoods.

Goodness of Fit Criteria

The performance of SRGM are judged by their ability to fit the past software fault data (goodness of fit) and predicting the future behaviour of the fault. The term goodness of fit is used in two different contexts. In one context, it denotes the question if a sample of data came from a population with a specific distribution. In another context, it denotes the question of "How good does a mathematical model (for example a linear

regression model) fit to the data"? To investigate the effectiveness of software reliability growth models, a set of comparison criteria is proposed to compare models quantitatively. Various criteria can be used to assess and compare the performance of selected models are described as follows.

SSE: The sum of squared error (SSE) is the total deviation of estimated failures from actual failures observed [19]. It can be defined as

$$SSE = \sum_{i=1}^{k} (m(t_i) - \hat{m}(t_i))^2$$

Adjusted R-square: The adjusted R-square statistic is generally the best indicator of the fit quality when two nested models are compared, (i.e. a series of models each of which adds additional coefficients to the previous model).

$$Adjusted\ R-square = 1 - \frac{SSE(n-1)}{SST(v)}$$

The adjusted R-square statistic can take on any value from 0 to 1, with a value closer to 1 indicating a better fit. Negative values can occur when the model contains terms that do not help to predict the response.

RMSE: The Root Mean Squared Error statistic is also known as the fit standard error and the standard error of the regression. It is an estimate of the standard deviation of the random component in the data and it is defined as [22]:

$$RMSE = s = \sqrt{MSE}$$

where MSE is the mean square error or the residual mean square

$$MSE = \frac{SSE}{v}$$

where v indicates the number of independent pieces of information involving the n data points that are required to calculate the sum of squares.

IV. CONCLUSION

In this paper we address the general concept of perfect debugging and imperfect debugging. We have discussed several existing NHPP SRGMs based on perfect debugging and imperfect debugging. We have discussed two parameter estimation techniques MLE and LSE which used to estimate parameters of different SRGMs. Finally we have discussed goodness of fit criteria used to assess and compare the performance of selected models.

REFERENCES

1. M. R. Lyu, "Handbook of Software Reliability Engineering". New York: McGraw Hill, 1996.
2. J. D. Musa,"Software Reliability Engineering: More Reliable Software, Faster Development and Testing," 2nd ed. Bloomington, IN: Author- House, 2004.
3. C. Y. Huang and C. T. Lin, "Analysis of software reliability modeling considering testing compression factor and failure-to-fault relationship," IEEE Trans. On computers, vol. 59, no. 2, pp. 283-288, feb. 2010.
4. C.Y. Huang and M. R. Lyu, "Optimal testing resource allocation and sensitivity analysis in software development," IEEE Trans. On Reliability, vol. 54, no. 4, pp. 592-603, Dec. 2005.
5. Jelinski, Z. and Moranda, P. (1972). Software reliability research. In Statistical Computer Performance Evaluation, Freiberger, W. (ed). Academic Press, New York, pp. 465-484.
6. 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. Software Engineering, vol. 29, no. 3, pp. 261–269, March 2003.
7. D. Kececioglu, "Reliability Engineering Handbook," vol. 2, Prentice-Hall, Englewood Cliffs, N.J., 1991.

8. M.Xie, Software Reliability Modeling. World Scientific Publishing, 1991.
9. Gaurav Aggarwal, Dr. V.K Gupta, "Software Reliability Growth Model," International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, Issue 1, Jan 2014.
10. 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.
11. H. Pham, "System Software Reliability." Springer London, 2006.
12. A.L. Goel, "Software Reliability Models: Assumptions, Limitations, and Applicability," IEEE Trans. Software Eng., vol. 11, pp. 1411-1423, 1985.
13. Kapur P.K., Garg R.B., "Optimal Software Release Policies for Software Reliability Growth Models under Imperfect Debugging", RAIRO, vol 24, pp. 295-305,1990.
14. Ohba M. and Chou X.M., "Does Imperfect Debugging Effect Software Reliability Growth?" in Proceedings of 11th International Conference of Software Engineering, pp. 237-244, 1989.
15. H. Pham, L. Nordmann, and X. Zhang, "A general imperfect software debugging model with s-shaped fault detection rate," IEEE Trans. Reliability, vol. 48, pp. 169–175, June 1999.
16. H. Pham and X. Zhang, "An NHPP software reliability models and its comparison," International J. of Reliability, Quality and Safety Engineering, vol. 14, no. 3, pp. 269–282, 1997.
17. H. Pham, "Software reliability and cost models: perspectives, comparison and practice," European J. of Operational Research, vol. 149, pp.475–489, 2003.
18. S. Yamada, K. Tokuno, and S. Osaki, "Imperfect debugging models with fault introduction rate for software reliability assessment," International J. Syst. Science, vol. 23, no. 12, 1992.
19. X. Zhang, X. Teng, and H. Pham, "Considering fault removal efficiency in software reliability assessment," IEEE Trans. Systems, Man, and Cybernetics—Part A, vol. 33, no. 1, pp. 114–120, 2003.
20. Brian S. Everitt and David C. Howell, " Least Squares Estimation," Encyclopedia of Statistics in Behavioral Science, 2005 , Vol. 2, pp. 1041-1045.
21. John Aldrich, "R.A. Fisher and the making of maximum likelihood 1912-1922," Statistical Science, 1997, Vol. 12, No.3, pp. 162-176.
22. W.K. Ehrlich and T. J. Emerson, "Modeling software failures and reliability growth during system testing," in proceedings of the 9th International Conference on Software Engineering (ICSE'87), Monterey, CA, March 30-April, 2, 1987, pp.72-82.

ACRONYMS

SRGMs	Software Reliability Growth Models
NHPP	Non-Homogeneous Poisson Process
GOF	Goodness-of-Fit
SSE	Sum of squared errors
RMSE	Root Mean Squared Error

NOTATION

$\alpha(t)$	the rate at which the faults may be introduced during the debugging process
$a(t)$	expected number of initial faults
$b(t)$	fault detection rate per fault in the steady state
$m(t_i)$	total number of failures observed at time t_i according to the actual data
$\widehat{m}(t_i)$	expected number of failures at time t_i estimated by a model