International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No.04, March 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

# Study of Imperfect Software Reliability Growth Models and Change Point Concept

**Shozab Khurshid, Javaid Iqbal**

*Deptt. of Computer Sciences, University of Kashmir, Srinagar, India*

**ABSTRACT**

*Non-Homogenous Poisson Process SRGMs that have been developed till now are based on the varying limitations. A number of Software reliability growth models (SRGMs) are built on the assumption that a fault responsible for software failure can be removed immediately without introducing any of the new faults. SRGMs built on this notion are termed as perfect debugging SRGMs. However, in reality the fault that is responsible for failure might not be removed perfectly or the removal of fault may give rise to some additional faults. This phenomenon is termed as imperfect debugging in the literature. Further, the rate at which the faults are detected during the testing process does not remain same. The moment of time at which the detection rate undergoes change is termed as change point. In this paper various imperfect debugging models have been studied. In addition to this, change point concept and models based on them are also discussed. Further, estimation techniques, comparison criteria and ranking of various Software Reliability Growth models is also discussed in our study.*

***Keywords*: Change Point, Goodness of fit, Imperfect Debugging, Mean Value Function (MVF), Non Homogenous Poisson Process (NHPP), Parameter estimation, Ranking, Software Reliability Growth Models (SRGMs).**

## I.INTRODUCTION

A very significant role is played by software in critical applications. Most of the real life functions can be automated through the use of software and thus their development and designing should be done in a very critical manner so as to prevent it from failures. Software system usually fails because of the fault that is present in them. Testing and debugging is done with the intent to detect and remove the residing faults in the software but it becomes impossible for software developers to launch fault free softwares. Whenever a fault is uncovered during testing, its removal process is carried out. Thus, a software that operates without failures always saves the precious time of its users and no inconvenience in operating the software is felt. This signifies that how much important it is to have a reliable software. Reliability of a software can be stated as the probability of its error free working under a certain environmental condition for some specified period of time [1].

Software reliability modelling considered as one of the subfields of software reliability engineering has received tremendous attention. Software reliability modelling helps us to measure the observed system reliability. To

quantitatively improve the reliability of the software products SRGMs are used. SRGMs estimate the reliability of the software and extrapolates the failure behaviour observed during testing to predict the future behaviour. Thus, SRGMs provide a significant help to software engineers during the testing phase and in the operational environment. SRGMs also provide a count of the leftover errors in the software. In reliability engineering, a poisson process is used to describe the fault counting. Mostly the failure count models are established on NHPP. This is because NHPP has a wide range of applicability, mathematical tractability and thus it is considered as one of the important classes of existing SRGMs [14]. NHPP models are used to describe MVF which denotes the cumulative count of failures upto a specific time period. NHPP models also describe the shape which the failure curve will take and it can be either concave, S-shaped or both. They have been successfully used in both hardware and software reliability in order to describe the growth of reliability [8]. A vast literature based on NHPP SRGMs has been developed over past 40 years. All these models vary on the basis of assumptions on which they are built. Some assumptions include that of perfect and imperfect debugging. In perfect debugging, no new errors are generated in the software on the removal of the faults that lead to the failure during testing. But in reality this is not the case and thus many SRGMs assume that the debugging environment is imperfect.

In this paper, a general idea of imperfect debugging environment is given and its effect on the fault content. The concept of change point has also been discussed. Further, techniques that are used for the estimation of the unknown parameters, fitting of the growth curve and ranking of the models is also discussed. The remaining paper is organized in the following sections. Section 2, discusses the types and some models based on imperfect debugging. Section 3, discusses the phenomenon of change point. In section 4, unknown parameter estimation techniques and various comparison criteria is illustrated. The significance of ranking has been discussed in section 5. Lastly, conclusions have been drawn in section 6.

## II Imperfect Debugging Environment and Its Types

Whenever a failure occurs, the testing team puts all its effort to remove the fault which resulted in this failure and the departure from the normal execution of the software. However, the detection team may be unable to remove the fault with perfection and either of the two cases will occur:

1. The original fault will remain as such.
2. The fault will get replaced by some other fault.

In the first case, the fault which seemed to have been fixed perfectly is actually imperfectly repaired. It is because the testing team finds the same failure on checking the code again on the same type of input. This phenomenon is known as (imperfect fault debugging) in which the fault content remains unchanged .In the second case, the error is removed but some other error giving rise to some other failure is generated. This phenomenon is known as (error generation) in which the fault content increases. Thus, imperfect debugging can be realized through two types [8] as shown in Fig. 1.

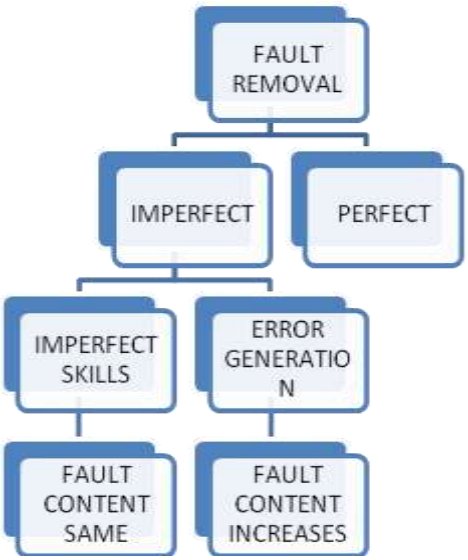


**Fig.1 Fault Removal Process**

Goel was first to give the concept of imperfect debugging. The effect of imperfect debugging was studied by him on the markovian model. The mathematical description of the imperfect debugging model classification and some SRGMs based on them is illustrated in the following sections.

### 2.1 Imperfect Fault Debugging Model

The assumption on which this model is based is that the faults are removed with probability $I$ which is considered as the perfect debugging probability, and with $1-I$ , the faults are removed imperfectly [5].

$$\frac{dm_f}{dt} = v(a - Im_f(t)) \qquad (1.1)$$

$$\frac{dm_r}{dt} = Iv(a - m_r(t)) \qquad (1.2)$$

Solving the above equations (1.1 and 1.2) we get

$$m_r(t) = a(1-e^{-vIt})$$

$$m_f(t) = (a/I)(1-e^{-vIt})$$

Where $m_f(t)$ and $m_r(t)$ is the MVF for faults detected and removed respectively. $a$ denotes the initial content of faults residing in the software and $v$ denotes the fault detection rate.

### 2.2 Error Generation Model

The first SRGM which incorporated error generation phenomenon was given by Ohba and Chou and was based on GO model. The error generation rate in this model was assumed to be a constant, $\gamma$ [6].

$$\frac{dm(t)}{dt} = v(t)(a(t) - m(t)) \qquad (1.3)$$

Where,

$v(t) = v$
$a(t) = a + \gamma m(t)$

Solving Eq. (1.3) with m(0)=0, the initial condition, we derive

$$m(t) = a/(1-\gamma)(1-e^{-v(1-\gamma)t})$$

### 2.3 SRGMs Based on Imperfect Debugging

Some commonly used imperfect debugging SRGMs of one or both the types are as follows:

1. Yamada Exponential Model [2]

In this model, an exponential function for testing effort is used. This model is concave in shape. MVF for this model is given as:

$$m(t) = a(1-e^{-v\theta(1-e^{-\beta t})})$$

Where $\theta$ is the total amount of testing effort and $\beta$ is the shape parameter.

2. Yamada Imperfect Debugging Model [3]

The concept of error generation is present in this model and it assumes introduction and detection rate of faults to be a constant. MVF is given as:

$$m(t) = a(1-e^{-vt})(1-\frac{\gamma}{v}) + \gamma at$$

3. Pham-Nordmann Zhang (PNZ) Model [4]

The assumption in this model is that the rate at which the faults are introduced is described by a linear function and the rate at which the faults are detected is an inflection S-shaped non-decreasing function. Its MVF is given as:

$$m(t) = \frac{a(1-e^{-vt})(1-\frac{\gamma}{v}) + \gamma at}{1+\beta e^{-vt}}$$

Here, $\beta$ denotes the inflection factor.

## III CHANGE POINT

It is because of the changes that are observed in the pattern of resource allocation, the environment in which the software runs, the strategies adopted by the testers etc. the detection rate changes. This phenomenon is known as change point and was given by Zhao[15]. SRGMs that incorporate this concept are said to represent the testing environment in a real sense. Tremendous SRGMs have been proposed by researchers that incorporate change point along with other factors[8,15]. One of the factor being that of perfect and imperfect debugging.

### 3.1 SRGMs Based on Change Point

1. Exponential SRGM[8]: The mean value function for exponential model based on single change point is given as:

$$m(t) = \begin{cases} a(1-\exp(v_1 t)) & 0 \le t \le c \\ a(1-\exp(v_1 c - v_2(t-c)) & c < t \end{cases}$$ Where $v_1$ and $v_2$ denote detection rates before and after change point $c$.

2. Weibull Model[8]: The distribution functions that are used for weibull model before and after change point are given as follows:

$$F_1(t) = a(1-e^{-\alpha_1 t^{\beta_1}})$$
$$F_2(t) = a(1-e^{-\alpha_2 t^{\beta_2}})$$

Where $\alpha_1, \alpha_2$ is the failure intensity function before and after change point and $\beta_1, \beta_2$ is the shape parameter before and after the change point.

### 3.2Estimation of Parameters and Comparison Criteria

After building a mathematical model it becomes very important for us to have estimation of the parameters that are unknown in the model. Actual failure data sets are used to estimate and validate these unknown parameters. The parameter estimation is usually done by solving an optimisation problem whose aim is to produce parameter estimates in such a manner that are very near to the actual, unknown parameter estimates. Non-linear functions are used to represent almost all the NHPP based SRGMs, the two most widely used methods for parameter estimation in such models are Maximum Likelihood Estimate (MLE) and Non-Linear Least Square (NLLS).An alternative method is to use SPSS(Statistical package for social science)in which minimum efforts

are required for parameter estimation This software has some predefined functions which are used for parameter estimation of the non-linear models. It is difficult to say that which one of the solutions either self-programming (MLE, NLLS) or the software (SPSS) can provide better estimates for the parameter.[8]

After selecting the model, the performance of the model can be seen by its ability to fit the observed data and how good is the model in predicting the future behaviour of the process. There are number of comparison criteria which can be used to compare the models quantitatively, in order to investigate their effectiveness. In this paper, we will discuss some of the comparison criteria used:

1. Mean-Square Error (MSE)[7,12]:The value given by MSE is the deviation between the observed data with the predicted values which can mathematically be shown as :

$$MSE = \frac{\sum_{i=1}^{k} (m(t_i) - Z_i)^2}{N}$$

Where N is the total count of observations, $m(t_i)$ denotes the predicted values and $Z_i$ denotes the actual values.

2. Bias[8,11]: Bias is given by the average of prediction Errors, which is the difference between the observations with the predicted values. Bias can be mathematically be defined as:

$$Bias = \frac{\sum_{i=1}^{k} (m(t_i) - Z_i)}{N}$$

Lower the Bias value, better fitting is achieved.

3. Coefficient of Multiple Determination( $R^2$)[9]:

$R^2$ gives the percentage of the total variation in account of the fitted curve about the mean. Lower percentage signifies that the fitting of the data with respect to the given model is not good. Mathematically, it is defined as the ratio of the residual sum of squares (SS) to corrected sum of squares subtracted from 1, and is given by:

$$R^2 = 1 - \frac{residualSS}{CorrectedSS}$$

4.The Theils Statistic(TS)[10]:

It gives us the percentage of the average total error with respect to actual values. Prediction capability of the model is good if its TS value is closer to zero.

$$TS = \sqrt{\frac{\sum_{i=1}^{k}(m(t_i) - Z_i)^2}{\sum_{i=1}^{k} Z_i^2}}$$

5. Root Mean Square Prediction Error (RMSPE) [11]:

The closeness with which the model predicts the observation is measured by RMSPE. It is mathematically defined as:

$$RMSPE = \sqrt{Bias^2 + Variation^2}$$

Where variation is given by the Standard deviation of prediction Error.[11]

### 3.3 RANKING

The main purpose of ranking is to select the best SRGM among all other SRGMs. Although comparison criteria can be used to select the best SRGM but sometimes it becomes difficult in case a SRGM shows better measure for some criteria and poor result for some other criteria. In that case, it is better to use ranking which will help us in giving accurate best model. There are many methods that can be used for ranking. Some of them are Distance based Approach (DBA)[13], Normalized criteria Distance[1] etc.

## IV.CONCLUSION

This paper gives a review of the imperfect debugging types, change point and some of the models based on them. Further in this paper ,the parameter estimation techniques and various criteria used for comparison are discussed. This paper also illustrates the ranking method and the importance of this method.

## REFERENCES

[1] M. R. Lyu, *Handbook of Software Reliability Engineering* (New York: McGraw Hill, 1996).

[2] H. Pham, Software reliability and cost models: perspectives, comparison and practice, *European J. of Operational Research, 149,* 2003, 475–489.

[3] S. Yamada, K. Tokuno, and S. Osaki, Imperfect debugging models with fault introduction rate for software reliability assessment, *International J. Syst. Science*, *23(12)*, 1992.

[4] H. Pham, L. Nordmann, and X. Zhang, A general imperfect software debugging model with s-shaped fault detection rate, *IEEE Trans. Reliability*, *48*, 1999, 169–175.

[5] P. K.. Kapur, R.B. Garg, Optimal software release policies for software reliability growth models under imperfect debugging, *Recherché Operationanelle/ Operations Research*, 24, 1990, 295–305.

[6] M. Ohba and X. M. Chou, Does imperfect debugging effect software reliability growth, In: Proceedings 11th international conference of software engineering, 1989, 237–244.

[7] H. Pham, *System Software Reliability* (Springer, 2006).

[8] P. K. Kapur, H. Pham, A. Gupta and P.C. Jha, *Software Reliability Assessment with OR Applications* (UK: Springer, 2011).

[9] 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*, 2008, 1410–1421.

[10] 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 Open BSD, in Proceedings of the 16th IEEE International Symposium on Softw. Reliability Engineering, Chicago, IL, 2005,193–202.

[11] K. Pillai and V. S. S. Nair, A model for software development effort and cost estimation, *IEEE Trans. Softw. Engineering*, *23(8)*, 1997, 485–497.

[12] S. Hwang and H. Pham, Quasi-renewal time-delay fault-removal consideration in software reliability modelling, *IEEE Trans. Systems, Man and Cybernetics-Part A: Systems and Humans*, *39( 1)*, 2009.

[13] K. Sharma, R.Garg, C. K. Nagpal, and R .K. Garg, Selection of Optimal Software Reliability Growth Models Using a Distance Based Approach, *IEEE Transactions On Reliability*, *59(2)*, 2010.

[14] R. Kaur and P. Panwar, Study of Perfect and Imperfect Debugging NHPP SRGMs used for Prediction of Faults in a Software, *IJCSC*, *6(1)*, 2015, 73-78.

[15] M. Zhao, Change-Point Problems In Software And Hardware Reliability, *Communications in Statistics-Theory and Methods*, *22(3)*, 1993,757-768.