ROOTINDEXING
JOURNAL ABSTRACTING AND INDEXING SERVICE
INTERNATIONAL ADVANCE JOURNAL OF ENGINEERING, SCIENCE AND MANAGEMENT (IAJESM)
January-June 2023, Submitted in February 2023, iajesm2014@gmail.com, ISSN -2393-8048
Multidisciplinary Indexed/Peer Reviewed Journal. SJIF Impact Factor 2023 =6.753

# Review of Literature on Modeling Software Reliability Growth from the Perspective of Imperfect Debugging

Jyoti, Former Assistant Professor, Government College Of Women, Bawani Khera, Bhiwani (Haryana)
Email- neer4ualways@gmail.com

## Abstract

Due to the fact that OSS is having free of cost access to the tools and technology, teaching and learning through OSS can be done in a wonderful manner. The gap of digital divide can be filled with the novel idea of OSS. Reliability measurement is a prime concern in OSS as it is being updated by many developers constantly. Research is being carried to develop SRGMs for OSS in order to check its reliability under different environmental conditions. Reliability of Mozilla Firefox, Apache, Genome etc. can be measured with the help of these SRGMs. The reliability models that have been proposed so far for OSS can be applied to reliability growth in particular and not in general. This is because of the reason that T&D environment is assumed to be different in different SRGMs. With the result, a particular SRGM is not applicable for any sort of environment. This has increased the need of formulating generalised framework for OSS.

**Keywords: Review of Literature, Modeling Software Reliability, Imperfect Debugging**

## Introduction

In today's fast moving life, almost everything is dependent on software systems. Software systems are developed with the intent to automate various real life functions of the most intelligent creature of the universe, the mankind. This dependence has increased the scope and importance of having highly reliable software in no time. The persistent and diligent research in the development of software systems has led to the innovation of some fabulous software products that has brought the mankind closer in order to share the experiences across a global platform. Multipurpose satellites, space shuttles etc. have been launched so as to forecast the things that are happening in the universe. Attempts are being made to explore places other than the planet earth for existence of life. However, to conquer such missions, highly advanced technology with high precision is required. Huge development costs are incurred by real-time and mission critical systems. On the other hand, high level of risk to human life is posed by safety critical systems. Thus, there should be no room for errors in the development of such systems. Even though the software system is created by the most intelligent creature of the universe, it is never failure free. The failures occur because of the faults that are manifested in them during their development by the software developers. The software testing team puts their best effort so as to remove the faults that are present in the software. However, the testing cannot be performed for long because of the stringent budget and schedule of the project management. On one hand, the project management wants all the faults that are residing in the software to be removed by the testing team so as to increase its reliability. On the other hand, the project management does not want to continue testing for long and increase the testing costs. Thus, scheduled delivery, cost and reliability are the main attributes for every software being developed. The main aim of the project management is to attain these attributes at their best possible values so as to achieve a good image in the market for long-term profits and survival.

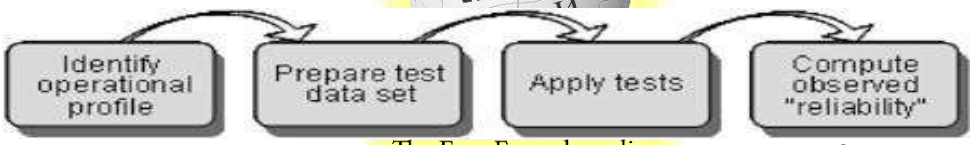
### Literature Review

The main aim of the testing process in the software development life cycle is to uncover all the faults that are lying dormant in the software. Software testing is defined as the process of executing a software system in its intended environment in order to determine whether or not the software matches its requirement specification. Dijiskstra (1972), states that software testing is an effective way to show the presence of underlying bugs inthe software and is not meant to show their absence. Whenever a failure takes place, the fault that is responsible for it is immediately repaired. The process of observing failure and removing the corresponding fault indicates that there is an improvement in the reliability of the system. Software

reliability being one of the most dynamic characteristic of software quality is preferred by both the users of the software as well as the developers of the software.

There are four types of testing methods viz. performance testing, defect testing, security testing, and statistical testing. Statistical testing is different from other methods of testing in the sense that statistical testing is used to measure the reliability of the software rather than uncovering the faults. It is considered to be the most effective sampling method for evaluating the reliability of the system and is also known as reliability testing. There are four stages in assessing the reliability of the software.

Reliability assessment provides both the users and the developers a quantitative measure of the leftover faults, decisions regarding the software release time, softwaremaintenance in the




operational phase etc. For users, reliability assessment provides a confidence measure in the quality of the software as well as their acceptability level.

Model proposed by Musa (1975) and the model developed by Musa and Okumoto (1984), also known as Logarithmic Poisson execution time model are the two most known models that lie in the category of execution time models. These models differ on the basis of underlying assumptions on which they are built.

Most of the SRGMs proposed so far, are based on calendar time, as this time component is more meaningful to the software developers, engineers and to the users of the software. A vast literature is available on calendar time models. In the year 1979, a pioneering attempt was done by Goel and Okumoto's model. The models that were proposed later aimed to incorporate various different aspects of T&D environment with the relaxation on some assumptions. Goel and Okumoto's (1979) model was exponential in nature.

Earlier, in NHPP modelling it was assumed that the failure process could be described by exponential models due to the uniform operational profiles. However, most of the testing profiles lack uniformity and thus the assumption of uniformity is not real. The testing profiles are thus non-uniform because of various different reasons.

Many researchers proposed models exhibiting S-shaped failure curve in order to model non-uniform testing profile. The S-shaped curve proved to be quite successful in describing the non-uniformity of the operational profile. A number of S-shaped SRGMs have been developed by many researchers.

Yamada et al. (1983) was the first to modify the GO model. They described testing as a two stage process, the fault-detection process and the fault correction process. Thus, the model proposed by Yamada et al. (1983) is known as Delayed S-shaped model. SRGMs proposed by Ohba (1984), Bittanti et al. (1988) and Kapur and Garg (1992) are also S- shaped in nature. However, these SRGMs have same mathematical form but they vary on the basis of assumptions on which they are built.



Depending on the values of the unknown parameters in the model, S-shaped models exhibit an important characteristic of describing both exponential and S-shaped growth curves. Hence are termed as flexible models. This flexibility makes S-shaped SRGMs more appropriate for real testing environments

**Types of imperfect Debugging**

In an imperfect debugging environment, Software Reliability Growth Models can be either purely imperfect, pure fault generation models while some others may integrate both the types of imperfect debugging. Goel (1985) first introduced the concept of imperfect debugging. He implemented it on Jelinski and Moranda model (1972). In thesetype of SRGMs, it was assumed that the removal rate of faults per remaining faults tends to decrease because of imperfect debugging. This is the first type of imperfect debugging phenomenon

The second type of imperfect debugging phenomenon is related to theerror generation. In this, the fault content by time infinity increases and is usually more than the initial fault content. The error generation phenomenon was described by Ohba and Chou (1989) in modelling SRGMs.

It is worth mentioning that during the early stages of research in reliability modelling, no distinction was made between the two types of imperfect debugging and even the models incorporating only one type of imperfect debugging phenomenon were simply named as imperfect debugging models. Thus, earlier a proper insight regarding this topic was not provided (Xie, 2003). The two types of imperfect debugging were first introduced by Zhang et al. (2003). The number of failures experienced/removal attempts were used in their modelling. A fault is generated only when some fault is being removed. Thus, the rate of generation of new faults is proportional to the rate of original fault removals. It should be noted that the number of failures that are experienced is not same as the number of fault removals. The facts related to imperfect debugging phenomenon were clearly illustrated by Kapur et al. (2006) in their model where they integrated both the types of imperfect debugging.

Another significant factor that plays a crucial role in evaluating the reliability of the software is testing effort. Testing effort is defined as the amount of the resources oreffort that are utilized during the fault detection/correction process in a software system. Testing effort is said to be directly proportional to the reliability achieved. Thus, software is said to obtain higher reliability if more resources are consumed during the testing process. However, due to the budget constraints, it is important to strike-off a balance between the resources utilized and the reliability obtained.

Numerous SRGMs have been proposed by many researchers that have incorporated the concept of testing effort (Ahmad et al., 2010a; Quadri et al., 2011; Kapur et al., 2012). Further, a unified model was proposed by Zhang et al. (2014) with testing effort under the imperfect debugging assumption. A SRGM was proposed by Li et al. (2015) in which the debugging environment was taken to be imperfect with S-shaped TEF being incorporated in the model.

Many times it is assumed that during the entire testing period, the parameters of the SRGM remain smooth. However, it is not always the case. For instance, after analysing the failure datasets after some days of testing, the management decides that there is a need of some additional skilled member to join the testing team and some changes are also brought in the strategy that was previously adopted for testing and even some advanced tools and techniques are employed for the testing process. These attempts are made in order to speed up the testing process. So, the parameters of the model before the changes were made will not be able to describe the testing process as some model parameters may undergo change. The kinks/jumps that are thus observed in the fault detection rate is termed as the change point. In the literature of regression, the term two- phase regression or multiple-phase regression is also used for change-point models. In addition to this, broken-line regression, switching regression, two-stage least squares or segmented regression is also used (Kapur et al., 2011a).

For hardware and software reliability, change point models play a very significant role. In software reliability modelling, it was assumed by most researchers that the fault detection rate remains constant and each and every fault has an equal probability of being detected. However, the detection rate of faults depends on testing effort, testing skills, size of the program and much more. Thus, the fault detection rate is not smooth and there is a possibility that it can change. It is very significant to incorporate the method of change-point

in order to analyse the reliability growth in the changing testing process. The SRGMs in which the change point effect is not considered in the estimation of software reliability is not the true representative of the actual testing environment (Zhao, 1993; Gupta, 2008).

In the process of analysing the change point, the studies that were conducted were related in estimating the change point position in case of a single change point, finding out the number of change points that are present and their positions if multiple change points exist and determining the parameters in case the distribution function between the change point remains same. Many authors have studied the problem of change point.

The reliability of a software can be assessed accurately with the change point phenomenon. SRGMs that are formulated by incorporating the change point method are considered to express the factual software reliability behaviour. As mentioned above, there are chances of no change point, only one change point and a number of change points depending upon the testing environment. Initially, Zhao (1993) carried out the studies for analysing the hardware and software reliability by incorporating the change point method. Later, a number of researchers proposed numerous SRGMs with the change point concept for measuring and predicting the software reliability (Chang, 2001;Huang, 2005; Shyur, 2003; Zhao, 1993; Zou, 2003).

SRGMs that have been proposed so far are built with diverse limitations considering different factors. Fault Reduction factor (FRF) is one of the factors that plays a very significant role in determining the reliability of the software system. Musa (1975) first identified the significance of FRF for determining the reliability growth.

In the process of testing, there is often seen some sort of relationship between the faults and the failures (Musa et al., 1987). When a user observes an unexpected software system behaviour, the failure is said to have occurred. On the other hand, data defined incorrectly in the software program or any other incorrect step results in a fault that further causes failure.

**References**

1. A. L. Goel, Software reliability models: assumptions, limitations and applicability. IEEE Trans. Software Engng SE-II, 1411-1423 (1985).
2. P. K. Kapur and R. B. Garg, Optimal software release policies for software reliability growth models under imperfect debugging. RAIRO 24, 295-305 (1990).
3. P. K. Kapur, Sanjay Agarwala and Said Younes, S-shaped software reliability growth model with imperfect fault detection. Private communication (1992).
4. S. Bittanti et al., A flexible modelling approach for software reliability growth. In Software Reliability Modelling and Identification (Ed. S. Bittani), pp. 101-140. Springer, Berlin (1988).
5. A. L. Goel and K. Okumoto, Time dependent error-detection rate model for software reliability and other performance measures. IEEE Trans. Reliab. R-28, 206-211 (1979).
6. Z. Jelinski and P. B. Moranda, Software reliability research. In Statistical Computer Performance Evaluation (Ed. W. Freiberger), pp. 465-497. Academic Press, New York (1972).
7. T. M. Khoshgoftaar and T. G. Woodcook, Software reliability model selection, a case study. Proc. Int. Syrup. Software Reliab. Engng, pp. 183-191 (1991).
8. J. D. Musa et al., Software Reliability: Measurement, Prediction, Application. McGraw-Hill, New York (1987).
9. M. Ohba, Software reliability analysis models. IBM J. Res. Dev. 28, 428-443 (1984).