Study on the Modeling Software Reliability Growth from the Perspective of Imperfect Debugging

[1]Dr. Gulshan Kumar and [2]Dr. Vivek Kumar

[1]Assistant Professor in Computer Science, Rawal Institutions,Sohna Road, Near Zakopur, Faridabad, Haryana (India)
[2]Assosiate Professor in Computer Science, Saharanpur Institute of Advance Studies, Saharanpur Uttar Pradesh (India)
Email- [1]Gulshan_dixit@rediffmail.com; viveks865@gmail.com

Abstract: 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.

[Kumar, G. and Kumar, V. **Study on the Modeling Software Reliability Growth from the Perspective of Imperfect Debugging**. *Researcher* 2025;17(3):11-16]. ISSN 1553-9865 (print); ISSN 2163-8950 (online). http://www.sciencepub.net/researcher. 03. doi:10.7537/marsrsj170325.03

Keywords: Modeling Software Reliability; Imperfect Debugging

Introduction

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.

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

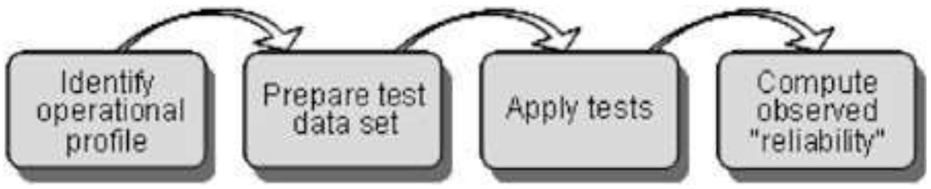
The main aim of the testing process in the software development life cycle is to uncover all the faults that are lying dormant in the software. Software testing is defined as the process of executing a software system in its intended environment in order to determine whether or not the software matches 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. 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

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 testingin 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, software maintenance 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

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 onthe basis of assumptions on which they are built.

Software has become one of the most significant components in day-to-day activity as the world is speedily moving toward technological age. Software firms put forth a lot of effort in order to develop free of faults software, thereby increasing the reliability of the software. Reliability modeling is done by software reliability growth models (SRGMs), which are software formulations used to evaluate and predict reliability.48 The modeling was started by researchers for single-release framework, and with time, it has been succeeded by multiple releases because of the demand for same. In the present scenario, software that has been incorporated with all the characteristics and is reliable but is formulated long back may turn out to be technologically disused. As a result, it is required to create software in more than one release/multi-releases where the new release may be the modification in existing feature with the aim to improve its reliability or addition in the functionality of software, etc.1, 2 It is not possible to detect all the faults in one cycle of software development as time and resources are accessible in a limited way. So, the remaining faults are rectified in the succeeding release, which can lead to imperfect debugging or error generation.1 Most of models assume that the rate of fault detection and/or removal stays the same throughout the testing.

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 debuggingphenomenon. 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 ofbeing 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.

Motivation

It is a must to assess the reliability of the software before it is deployed in the market. The customer satisfaction and acceptance are very unrelenting whereas on the other hand there is not too much flexibility in the testing costs and the scheduling deadline of the project. As a result, there arises a need to make a balance between these two (Kapur et al., 2011a). Even after employing skilled testing staff, upgrading the testing tools and techniques, using testing resources in an optimized manner etc. so as to minimize the testing costs and release the software in the market on time, the management fails to understand the practicality involved behind the testing process. The process of fault detection, isolation and correction, the effort utilized in removing the faults, the fault dependency, the severity of faults, the time lag involved in the detection process and the correction process etc. makes the job of testing team bound to take time and involve costs. The decision on the release time of the software by the management is determined by considering the reliability growth versus the cost involved in the testing process. Number of ways are available to evaluate the reliability of the software. Software Reliability Growth Models (SRGMs) is one of the most popular methods among all the available methods. The most important part in building these SRGMs is the ability to perceive the real environment and then effectively model them as assumptions under some given analytical framework.

These SRGMs are able to represent the testing process by taking into consideration perfect/ imperfect debugging modelling, learning effect modelling, testing-effort modelling, change-point modelling, FRF modelling so as to effectively incorporate the actual/real behaviour of the software development process by making use of state-of-art methodologies, strategies and tools. Many industrial environments and development processes make use of these SRGMs. In order to build confidence in the amount and levels of testing conducted on safety-critical and mission-critical applications, some of these SRGMs have been selectively used. The SRGMs that have been proposed so far covers only some aspects of the T&D (Testing and Debugging) environment and thus can be applied to some particular reliability growth phenomenon. This study proposes SRGM that is generalized in nature, covers many aspects of T&D environment together and has the capability of obtaining several different SRGMs.

Non-Homogeneous Poisson Process

Since years, reliability modelling has been used to formulate models that can be used to represent and analyse the operation of the real testing environment. There are two main types of processes that are used for modelling stochastic processes: continuous and discrete. Counting process which is present in the class of discrete stochastic processes is used for reliability engineering. A counting process is defined as the process which is used to describe the occurrence of some event (e.g. occurrence of a software failure, a software repair etc. In reliability engineering, poisson process is most widely used in order to represent a counting process. To describe the reliability growth and the deteriorating trends in hardware reliability, non-Homogeneous poisson process is used. With this, a number of NHPP based SRGMs have been proposed by many researchers following the trends in hardware reliability. With the help of SRGM, one can describe the process of failure occurrence and fault removal phenomenon with respect to time. The unit of time taken can either be clock time, calendar time, execution time or even test cases

Properties of NHPP Software Reliability Modelling

• The shape that can be taken by the failure curve in NHPP-based SRGMs can be either concave, S-shaped or a mixture of two. Thus, NHPP-based models can be categorized into two broad categories: Concave models and S-shaped models.

• The failure pattern in concave models exhibit an exponential shape.

• In both NHPP-based models, it is seen that as the number of detected faults increases with time, the fault detection rate starts decreasing and tends to approach some finite value. Thus, these models have same asymptotic property.

• The learning effect during T&D process is described by the S-shaped models. The S-shaped failure pattern describes the initial process of testing to be less efficient as compared to the testing performed in the later stages.

Concluding Remarks

A brief overview of the chosen area of research is given with emphasis on Software Reliability Engineering (SRE). The motivation which is behind this research work is also discussed. Moreover, the various objectives to be achieved in this research study are also presented. After this, the contributions that are made in this research are presented. Finally, the chapter concludes with the organisation of the remaining chapters of the study.

References

Aggarwal, A. G., Tandon, A., & Nijhawan, N. (2015). 'A Change Point Based Discrete SRGM for Multi-Release Software System', In: Published in the Proceedings of International Conference on Evidence Based Management (ICEBM), BITS Pilani, pp. 674-678.

Ahmad, N., Khan, M.G.M., & Rafi, L.S. (2011). 'Analysis of an inflection S-shaped software reliability model considering log-logistic testing-effort and imperfect debugging', International Journal of Computer Science and Network Security, vol. 11, no. 1, pp. 161–171.

Bittanti, S., Bolzern, P., Pedrotti, E., Pozzi, M., & Scattolini, R. (1988). 'A flexible modeling approach for software reliability growth', In: Goos G, Harmanis J (eds) Software reliability modelling and identification, Springer, Berlin, pp. 101–140.

Bokhari, M. U., & Ahmad, N. (2006). 'Analysis of a Software Reliability Growth Models: the Case of Log-logistic Test-effort Function', in: Proceedings of the 17th IASTED International Conference on Modeling and Simulation, Montreal, Canada, pp. 540-545.

Chiu, K. C., Huang, Y. S., & Lee, T. Z. (2008). 'A study of software reliability growth from the perspective of learning effects', Reliability Engineering and System Safety, vol. 93, issue 10, pp. 1410–1421.

Dijkstra. (1972). 'Disciplines of Programming', Prentice Hall.

Garmabaki, A. S. H., Barabadi, A., Yuan, F., Lu, J., & Ayele, Y. Z. (2015). 'Reliability Modeling of Successive Release of Software using NHPP', Industrial Engineering and Engineering Management (IEEM), IEEE International Conference, pp. 761-766

Hamilton, P. A., & Musa, J. D. (1978). 'Measuring reliability of computation center software', in: Proceedings of the Third International Conference on Software Engineering, pp. 29–36.

Hsu, C. J., Huang, C. Y., & Chang, J. R. (2011). 'Enhancing software reliability modelling and prediction through the introduction of time-variable fault reduction factor', Applied

Mathematical Modelling, vol. 35, no. 1, pp. 506–521.

2/5/2025