JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p.941-946
https://publishoa.com
ISSN: 1309-3452

Novel Techniques for Software Reliability Prediction

Dr. M. JAHIR PASHA[1], K. SWETHA[2], MD ASIM[3]

[1] Professor & Head , Dept. Of CSE, Ashoka Women's Engineering College, Kurnool, AP.
[2] Assistant Professor, Dept. Of CSE, Ashoka Women's Engineering College, Kurnool, AP.
[3] Assistant Professor, Dept. Of CSE, Ashoka Women's Engineering College, Kurnool , AP.

ABSTRACT
With the rapid advancement in computer technology and quick development of computer applications, the scale of the software system has enhanced and the function has become more complicated issues. Hence, the requirement of software quality is in great demand. In general terms, the probability failure free operation is termed as software reliability for a particular duration of time in a particular environment is one of the major qualities metric. Over the past decades, software has become an important solution for every purpose from elementary basic education to scientific studies and other research. Necessity and dependency for computer increases the software failures. The increase in number of computer users‟ also leads to the difficulties in maintaining the software requirements. To avoid these problems software reliability essential to adapt at the time of development of software so that they can get solution for software reliability that can address these issues. Software testing is the important issue to be considered in quality control used while doing the software development.

Keywords: Software Reliability, software testing,Software requirements.

1. Introduction

Maintainability, portability, usability, security, reliability and availability are the quality of the software system attributes. The reliability of software is also the important attribute, they can be evaluated and predict the operational quality of the software products. The main objective of software reliability professionals is to increase the reliability of the software that a specified program should work as per the instructions by the hands of the users. The general rule of accepting the metric for quantifying a product‟s reliability is the number of failures one can expect to find within certain duration. Faults lead to the mistakes in the codes of software and many faults may come out from of one fault. The steps of evaluating and eliminating failures to improve the performance of the reliability of software will be expressed mathematically. These expressions are known as SRGM. Improvements of the software reliability and detecting software faults can be done by applying mathematical modeling. Hence, SRGM is used to stop the testing during the attainment for the given reliability level. Mainly two important categories of software reliability models which are used to estimate the software reliability from design parameters and test data.

Software Reliability Growth Model (SRGM) has been developed during the last 30 years and they can yield very useful details and information about the improvement of reliability quality. From For the past 30 years, variety of statistical models has been proposed to access the software reliability procedures. The most general way of approaching the development of software reliability model is the probabilistic method.

Probabilistic models are divided into various they are as follows

- Markov structure models
- Non Homogenous Poisson Process (NHPP)
- Error seeding models,
- Failure rate models,
- Curve fitting models
- Reliability growth models.

Best Practices for Software Testing

In order to shutter the current challenges facing software testing, several actions needed to be taken. This section presents testers best practices that are capable of resolving the current issues in software testing. Testing process consist of a test plan to a collections of test cases known as test suite. A well defined test plan and user requirements are very important to the software development community. Wrongly defined and develop the test plan and the requirement causes huge impact to the organization.

It is presumed that software tester and development members should be a part of requirement gathering team for better understanding of application requirements. The persons must be trained in such a way that, they should have the deep knowledge and skills ranging from requirements analysis to systems design and programming skill. Gillenson, Racer & Richardson found that person

must be well versed in a breadth of skills ranging from requirements analysis to systems design and programming to be competent software tester; tester should possess a broadly based systems analysis skill and back ground that exhibits the necessity of high level systems design from one to one and on the other.

Understanding of the software product under testing is very crucial trait qualified software tester should have. In order to analyze domain knowledge divided the software tester domain knowledge into two important views which includes the users‟ point of view and application domain point of view respectively. They claimed that users‟ perspective includes knowledge of the real time process of actual use and actual users of the software system combined with a good understanding of the actual operational context of the system. This must be an inclusive of the higher level knowledge requirements and aim of the customers.

That means, for what purposes the system serves as part of the customers own purposes. Hence, they assumed that as the knowledge category.

Software Development Process

There is urgent requirement to upgrade software reliability by removing the faults or errors which accrued during the development of software codes. The academic institutions and industry have largely responded to this requirement by enhancing the development techniques in the name of software engineering and their by employing regular testing for finding faults in programs of software during the development.

There is urgent requirement to upgrade software reliability by removing the faults or errors which accrued during the development of software codes. The academic institutions and industry have largely responded to this requirement by enhancing the development techniques in the name of software engineering and their by employing regular testing for finding faults in programs of software during the development.

New programs are composed by altered the original code by comprised more of a bias near statements that arise in pessimistic execution paths. For that purpose in proposed method contains fault localization information to indicate the position of fault.

In experimental as well as regression based equations represent the soft computing techniques results is better compare to the other techniques. Evaluation of soft-computing techniques represented that accuracy of the ANN model is superior to the other models. Data bases for performing the training and testing stages were collected, these soft computing techniques had low computational errors than the empirical equations. Finally says that soft computing models are better compare to the regression models. We know that, when the electronics products delivered from the distributors, we do not know the destination of these products.

Hence, finding faults and correcting a serious software problem would be better instead of recalling thousands of products, especially in automotive sector.

Software reliability assessment is the critical part for understanding the significance and distinction between fault and failure. In the software, program code itself is faculty, it may not meet the customer or user standard and it can be considered as failure. Thus, code is the main defect that leads to failure.

Performance Measures of Model Parameters

SRGM success mainly reliable by gathering the accurate failure information. The functions of the software reliability growth model were predicted in terms of such information gathered only. To evaluate this technique and to equate its performances with other existing models, experiments can be conducted on actual software failure data. There are two important techniques used to estimate the model parameters for this purpose they are as follows.

- Maximum Likelihood Method (MLE)
- Least Square Estimation (LSE) models

Least Square Estimation Model (LSE)

The parameters α ,β ,m,δ and NMWTEF will be predicted by LSE method. These functions are calculated for *n* data points which are observed and can be expressed as,

$(g_k,X_k)(k=1,2,3\ldots\ldots n:0<t_1<t_2<t_3\ldots\ldots<t_n)$

where, α ,β ,m,δ can be computed by minimizing the equation given below.

$$S(\alpha,\beta,m,\delta)=\sum_{k=1}^{n}[X_i-X(t_k)]^2$$

Maximum Likelihood Method (MLM)

If the functions of α,β,m,δ are known, these functions of the SRGMs will be predicted via Maximum Likelihood Estimation method.

The values of a and r are determined for n observed data pairs in the form $(q_{k,}y_k)(k=1,2,\ldots n:0<t_1<t_2<t_3\ldots.<t_n)$,

where $y_k =$ Cumulative number of software faults detected up to time q_k or $(0,q_k)$. Then the likelihood functions for the unknown parameters a and r are in the SRGM can be written as,

$$L=\prod_{k=1}^{n}\frac{[m(q_k)-m(q_{k-1})^{(m_k-m_{k-1})}}{(m_k-m_{k-1})!}e^{[m(t_{k-1})]} \tag{1.3}$$

where, $m=0$ and $t=0$. This equation can be computed by mathematical modeling method to obtain the values of a and r.

Mean of Squared Errors (MSE)

The MSE can be mathematically expressed as,

$$MSE=\frac{1}{k}\sum_{i=1}^{i=k}[m(t)-m_i]^2 \tag{1.4}$$

where, $m(t)=$ Expected number of errors at time t_i estimated by a model

$m_t =$ Expected number of errors at time t_i. MSE gives a quantitative comparison for long term predictions.

Relative Error

The Relative Error (RE) can be mathematically expressed as,

$$\mathrm{Re}\,lativeError(RE) = \frac{m(t_q) - q}{q} \qquad (1.5)$$

Assume that we have noticed that, at the end of test time t_q, there is a failures q. It uses the failure data up to $t_q (\le t_q)$ for determining the parameters of $m(t)$.

Testing-Effort Function for modeling SRGM

In software testing phase, the significant test effort is necessary for human power, CPU time and number cases involved. After installing the software on computer, it is needed to be updated frequently to locate any problems or errors found in the software. The program updating can be done using the software patches present inside the package. Once updates installation got over, any problems or drawbacks in the program will not occur at all. Need for software development: As the present-days are highly competitive, each and every companies and organization has been looking for change over, cost conscious, high quality and design, scalable and customized solutions, which helps in functioning speedily and to provide quick results. To co-ordinate this function, many recognized and government organization is having the influence of edging methods to give the organizations, an appropriate or suitable solution to satisfy the needs of them. Suitable customized software development has now become the object for producing a good solution.

We all know that, the test effort during the testing phase and the time dependent behavior of development effort in the software development process can be described by a Weibull type consumption curve.

It is found that, these testing effort curves which are not smooth during the process of software development. Hence, a LTEF instead of the Weibull-type testing effort consumption functions as the testing effort function to illustrate the test effort patterns during the development of software.

Applications of Software Reliability Test Models

Software Reliability Growth Models (SRGM) helps us to show number of defects or failures that may occur after the completion of a software product and give us indication whether the software is ready for use or not. Different models have different ways to predict the failures. Few of these methods use system data to predict the number of faults in the software. Most of the software reliability growth models have functions that relates to the entire of faults present in a set of programme codes. This model fails to work smoothly when there are some issues left during the previous stage. Thus, this model fits well for the developers who already have designed and built similar software in the past and those aware of its entire domains.

If these functions and the number of faults are known, we can find out how many number of faults are remaining. Some of the applications of SRGM are as follows;

- The S-shaped software reliability model is applied to real time systems to detect errors accurately.
- The user oriented software reliability teaches the user how to effectively analyze the system reliability based on statistical analysis which detects fault.
- The reliability models are used to find very good strategic testing techniques to make testing process more effective.
- The testing strategies make our system more reliable.
- The reliability model identifies the critical modules so we can identify and correct the modules before failure occurs.
- The user oriented reliability model is a applicable to both software and also hardware.
- The software reliability growth models are used to indicate error, maintenance, cost minimizing process.
- SRGM is applied to data set.
- SRGMs are applied in software project management.
- The SRGM models are used to analyze the quality of service provided to the customers.

Challenges in Software Reliability Tests

Testing of software today is a huge challenging activity in software system projects. The following are the some of the current challenges.

- Less Competent Testing Personnel

- Lack of Testing Model
- Time Consumption
- Spurious Warning and Getting a Suitable Set of Cases
- Maintenance of Testing as a Service
- Poor documentation
- Large number mathematical models with different types were available in the literature
- In general, the software development process is sensitive and it should be kept secret because of the huge competition in the software industries.
- If software developer's releases more sample data sets, more improvement in developing useful software growth models.
- There will be considerable amount of time delay between finding a software fault and resolving or fixing the faults.
- Because of interlocking nature of faults and diverse, modeling of the failure process in the part of the software is big challenging task.

2. Methodology

Due to technology development, we rely on software intensive systems for our day to day life; there arises the need for soft ware reliability. The term "Software Reliability" defines the error free software for specified period of time in a specified environment. Measurement of software reliability plays an important role in optimization of software development and maintenance cost.

Software reliability represents the probability of failure-free software operation for a certain period of time in a particular environment. Over the past 30 years, lots of SRGMs have been developed and most SRGMs presuppose that detected faults are corrected at once. In literature, many software reliability growth models were utilized and the parameters related to software reliability were estimated. Besides the other, the most recently developed SRGM for distributed environment incorporating two types of imperfect debugging method estimates the software reliability in two types of software sub systems. The software sub-systems are reused system with simple faults, and newly developed system with hard faults and complex faults, respectively.

The Mean Value Function (MVF) of all these systems were computed separately and summed to estimate the software reliability. However, this technique has drawbacks in fault consideration process. The software systems basically have independent and dependent faults in the debugging process. The existing SRGM has focused only on the fault severity. It will not find whether the faults are independent or dependent to the system. The process software testing is processed while coding is developed by programmer and full system test is taken by some experts in testing at different levels of program code like testing of the module, program, product, in-house and the product at user's end.

The lack of such fault consideration in imperfect debugging process does not produce accurate estimation results.

If the aforesaid drawbacks in the literary works are solved, then the SRGM imperfect debugging process will be improved with high accurate results. Hence, the lack of solution for such limitation has been motivated to take up this research work.

Software reliability is similar to hardware reliability but finding software errors are difficult than finding hardware errors .Software reliability depends on the reliability of design involved to build a system , mapping that design to implementation and reliability of software components involved. The general expected problems in debugging of software systems are listed below.

Nature of software/ Middleware: As in the name, middleware "sits" between applications and platform libraries leading to lot of third party dependency in debugging issues. Often the features spread across the stack from applications to platform software, and hence isolating problematic module can be challenging.

Debug “unaware” Architecture: A common mistake in software architecture is ignoring the debugging aspect. For example, a too much multi-threading design may ease data transfer and communication, but it can end up causing stability concerns and toughness in debugging. Lack of adequate debug interface to monitor the software at run time can cause long term inefficiency in debugging.

Complexity in reproduction of bug: Unlike applications which can use simulators and controlled environment like desktop to reproduce and analyze the bugs, most software doesn’t have that luxury, as most software programs are developed for embedded products and involves platform abstraction.

Lack of systematic approach: When the developers are wayward in their approach to the bugs, they normally take long time to spot the issue or even worse ends up identifying a wrong reason as cause. The latter is even worse as it can not only cause side effects, but also an expensive QA test cycle and delay in deployment.

Not using automation in test cycle: For hard bugs such as stability bugs, the manual reproduction can be tough and may take longer waiting period for logs. The same cycle also goes for verification part as well.

3. Conclusion

In this paper there are three SRGMs which have different approaches. In the first work, two types of techniques were considered to unify the broad range of reliability growth models for software testing dependent upon testing effort under a general modeling architecture have been discussed. Even though these models were derived from different set of assumptions, they are proved to be equivalent mathematically. This work also integrates one of the testing efforts function based models.

References

[1]. Musa J D, Iannino A, Okumono K, “Software reliability, measurement, prediction and application,” New York: McGraw Hill; 1987.
[2]. T. M Khoshogoftaar, T. G. Woodcock, “Software reliability model selection: a case study,” Proceedings of the International Symposium on Software Reliability Engineering, pp. 183–191, 1991.
[3]. S. Bittanti, P. Bolzern, E. Pedrotti, R. Scattolini, “A flexible modeling approach for software reliability growth,” Springer Verlag, Berlin, 1998, pp. 101–140, 2003.
[4]. M. Ohba, “Software reliability analysis models,” IBM J. Res. Dev, pp. 28428–443, 1984.
[5]. C. Y. Huang, C.-T. Lin, “Software reliability analysis by considering fault dependency and debugging time lag,” IEEE Trans. Reliability, 55(3), pp. 436–450, 2006.
[6]. Brocklehurst S. and Littlewood B., “New Ways to Get Accurate Reliability Measures,” IEEE Software Transaction, vol. 9, no. 4, pp. 34-42, 1992.