Journal of Advances in Science and Technology
Vol. 17, Issue No. 1, March-2020, ISSN 2230-9659

A Study of Software Reliability Growth Models

Sharad Kumar Dubey[1]*, Dr. Rajeev Yadav[2]

[1] Research Scholar, Shri Krishna University, Chhatarpur M.P.

[2] Professor, Shri Krishna University, Chhatarpur M.P.

Abstract - The main objective is to present a classification of reliability models that would be useful in determining which of the existing model to use in given software development environment in this study importance is given on comparison of existing software reliability models. The analytical models are mostly useful in estimating and monitoring reliability. The models can help software testing/debugging managers to make predictions about the anticipated future reliability of software under growth and the study in which discussed about summary of NHPP based continuous time software reliability growth models, NHPP based software reliability growth models , development history of software reliability growth model, software reliability models classification.

Keyword - software, models

----------X----------

INTRODUCTION

Software reliability is a critical component of computer system availability, so it is important that Tandem's customers experience a small number of software failures in their production environments. Software reliability growth models can be used as an indication of the number of failures that may be encountered after the software has shipped and thus as an indication of whether the software is ready to ship. These models use system test data to predict the number of defects remaining in the software. Software reliability growth models have been applied to portions of several releases at Tandem over the past few years. This experimental research has provided some insights into these models and their utility. The utility of a software reliability growth model is related to its stability and predictive ability. Stability means that the model parameters should not significantly change as new data is added. Predictive ability means that the number of remaining defects predicted by the model should be close to the number found in field use. [1]

Reliability along with reliability, flexibility, efficiency, serviceability, capability, installation capacity, maintenance capacity and documentation are an essential feature of software quality. The software reliability is described as "the possibility of failure-free operation of software over a specified period of time in a specified setting" according to ANSI (American National Standards Institute). Computer Software Trustworthiness is difficult to obtain since software is also highly technical. While it's impossible to achieve a certain degree of reliability with all extremely complicated structures, like applications, device engineers prefer to move complications through their software layer, with the quick growth of systems size and simple to do so with a software update. Although software complexity is inversely associated with software stability, it is directly linked to other major software quality variables, especially functionality, capacity etc. These functionalities serve to make the app more complicated.[2]

Software Reliability Models Classification

The Reliability Model for Software Reliability (SRGM) is the instrument used in objective software evaluation, test status, time scheduling, and reliability improvements monitoring. In the last two decades, a range of software reliability models have been established. The rest of these are historical evidence obtained during the examination phase dependent on loss. These models were used to assess program consistency and potential trustworthiness forecasts. They were often seen in several management decision-making issues during the research process. But neither of these models may say that they are the best and so more study is needed. This segment provides a short overview of various modeling approaches.[3]

(a) Classification Schema

The versions are largely known as black boxes (Single System) and White boxes (Software for Multi Components). The black box models are further divided into Inter Failure Periods, Failure Count, and Static Models. Markov also presupposes this framework which correlates with the Failure Count and the Inter Failure Period Modelling. The Bayesian model may be generalized to include all existing models. We estimate models using Bayesian techniques if the model is Bayesian. White box versions are the models for modular device applications that take into consideration the system

design. The models for the white box are commonly categorized in state, path-based and additive models. Continual Time Markov Chains (CTMC), Discrete Time Markov Chain (DTMC) and Smi-Markov models are all classified in the State Based Scheme. All of these model Markovs collapse either into Markov chain class absorbent or irreducible. Absorption style implies terminating program, and the operating system constantly consists of an irreducible chai.[4]

(b) Single System Software

The structure of the model is not taken into account while estimating the dependability of the software system in this approach, which treats the whole programme as a single monolithic system. Reliability is based purely on the past failures of the product. Models like Goel-reliability Okumoto's estimate are popular. Both the Goel and Okumoto (1979) as well as the models.[5] Because these models do not take into consideration the structure of software, for example, they are not as accurate. It is common to predict the behavior of single-system software failures using the Software Reliability Growth Model and classify them according to the time elapsed between failed operations and the number of failed operations.

(c) Failure Rate Models

The most important parameter in these models is the period between failures. As more flaws are discovered and fixed, it is predicted that the time between failures will grow. Because inter failure times are unpredictable and prone to statistical fluctuations, this may not always be the case. It is impossible to decrease mission time's dependability function in any way. This shows that the software's credibility has risen.[6]

(d) Markovian Models

To describe the failure process in terms of a markov chain is known as a Markovian model. Depending on the amount of defects still present or problems previously eliminated, the software might be in a variety of states at any one moment. There are two factors that influence the likelihood of a transition: the software's present state and its transition probability. The exponential distribution of failures in the Markovian process is a result of its memory less nature. As a result of the well-developed idea of hardware dependability, several efforts have been made to build Markovian models. [7]

(e) Models based on Bayesian Analysis

The unknown parameters of the models are estimated using the least squares approach or the maximum likelihood method in the preceding two categories (later in this chapter both these methods are briefly discussed). However, the Bayesian analysis method is employed to estimate the models' unknown parameters in this category. This method makes it easier to utilize data gathered throughout the development of comparable projects. Given this data, it is reasonable to infer that the model's parameters will follow some kind of distribution (known as priori distribution). A posterior distribution may be calculated from the software test results based on the a priori distribution, which then represents the failure phenomena. [8]

(f) Static Models

Only if all failure data is accessible can these models be used to assess software dependability. These models were created at the beginning of the development of the dependability model and are now seldom used since they cannot take into account the software structure.

(g) The Input Domain and Fault Seeding Models

It is common in fault seeding models to seed the programme with a predetermined number of flaws. Finally, the programme is put through its paces. Inherent and seeded flaws were found in the observed bugs. Maximum likelihood estimate and combinatory are used to find the inherent and seeded flaws in the programme, and the resulting number of flaws is then determined. Seeded faults and intrinsic defects must have equal detection probabilities in order for this strategy to work. Getting there is a challenge. An input distribution is used to produce a collection of test cases in an input domain-based model. From the number of test failures seen during symbolic or physical execution of the sampled test cases, the reliability metric is derived. The representative input area is used to choose the test cases, which are then run and the results are recorded. Statistical methods may be used to assess the likelihood of success. The input distribution (operational profile) is notoriously difficult to predict; typically, the input distribution is derived from the many software routes.[9]

(h) Software Metrics Models

In this class of models, the fault content in the software is correlated to the length, complexity, and volume of the programme. Because these models are based on actual evidence, their output is highly reliant on the particulars of the software development process, which may differ from project to project.

(i) Multi Component Models

Models like Multi-Component White Box take into account the system's Software Architecture and its interactions with its many subsystems. Component-based software is modeled using white box models. It is the goal of such models to clearly reflect the testing methodology and the software architecture utilized throughout the testing phase. [10]

(j) State Based Models

It is assumed in architecture-based models that a component failure will lead to a system failure in the long run. Software components in State Based Models need a utilization factor, as opposed to hardware components, which are constantly in use.

(k) Composite-Hierarchical Approach

Based on the solution technique, the State Based Models are further subdivided into composite and hierarchical. The architectural model is combined with the failure model and then solved for reliability prediction using the composite approach. As a result, it is possible to estimate dependability of architecture by combining a failure model with an architecture model solution.[11]

(L) Path Based Models

Like State-Based Models, Path-Based Models take into account the components and interfaces that make up software architecture. Experimental or algorithmic methods are used to discover the various routes across the system. The dependability of a route is the sum of the reliabilities of each of its components. It is the average of the path reliabilities that make up a system's dependability. Infinite loops in a route may be accounted for using state-based models, while path-based models end the loop at one or the path's average execution time. [12]

(m) Additive Models

Each component's dependability is modeled using the NHPP in addition to the software testing phase. System failure rate is NHPP, which means that the cumulative number of failures and their intensity functions are sums of the function of each component. The architecture of the programme is not considered in the additive model. Song, Kwang& Changcreated an architectural model based on additive geometry. Markovian assumption of exponential failure and repair durations is relaxed in a restricted way by Semi Markov and Markov regenerative models. There is a state space explosion issue that affects them as well In contrast to analytical models, discrete-event simulation is an intriguing option since it can capture a comprehensive system structure and assist the investigation of many elements such as reliability growth and various repair procedures. It's a simple issue that all software development teams must answer: when to cease testing and release the product, whether the software is a single component or a multi-component application.

NHPP Based Software Reliability Growth Models

The NHPP models have the anticipated amount of defects/fails. For the definition of the behavior of a device over time, stochastic processes are used. Stochastic systems are of two major types: continuous and discrete. Among discrete methods, reliability engineering counting processes are commonly used to characterize events in time (e.g., failures, number of perfect repairs, etc). A Poisson method is the shortest counting procedure. In reliability engineering they are particularly important as a general class of the well-developed stochastic process model in describing failure processes which have certain patterns, like reliability growth and decline. Thus, the usage of NHPP models is conveniently enforced for device usability research. The models show the predicted amount of defects/fails at a certain moment. Each SRGM is focused on a variety of assumptions that are appropriate for a given research setting. The abundance of SRGM is causing the selection dilemma. In a model collection, the most relevant parameter is the feasibility and the importance for the actual test setting of model assumptions. In the presence of several SRGMs, the issue in model selection is a repetitive activity. In addition, two additional main parameters are the success of a model in terms of its capacity to recreate previous failure data and to forecast the future of the failure observation phase.[13]

Summary of NHPP Based Continuous Time Software Reliability Growth Models

A considerable number of continuous time models in the literature have been created to track the phase of elimination of errors and to calculate and forecast information systems' reliability. The association between the test period and the related amount of deleted errors was observed during the testing stage either exponentially or S-shaped. Another type of models is accessible, known as versatile models.

These models will show both the exponential and the S-shaped phenomena of failure progression, depending on their parameter values. Any of the well-known styles are below.[14]

- Model due to Goel and Okumoto (1979) (purely Exponential)
- Model due to Yamada, Ohba and Osaki (1983) (purely S-shaped)
- Model due to Ohba (1984) (Flexible)
- Model due to Bittanti at al. (1998) (Flexible)
- Model due to Kapur and Garg (1992) (Flexible)
- Generalized SRGM due to Kapur, Younes and Agarwala (1995) (Model for fault of different severity)

Development History of Software Reliability Growth Model

Software reliability is an important branch of software engineering and a new technology. In 1985, the TC56 Technical Committee of the

International Electro technical Commission established the Software Reliability Working Group and began to develop standards for software reliability. Software reliability has grown considerably over the next 20 years. An important branch of software reliability is the study of software reliability evaluation. Data through software reliability evaluation can better control resources, enable software to be distributed on time, and meet user reliability standards for software, reducing costs and increasing revenue. In the past 20 years, a large number of software reliability growth models have been proposed. The non-Homogeneous Poisson Process (NHPP) software reliability growth model is the most influential one in the reliability growth model. [15]Researcherfirst proposed the software reliability growth model (JM model). In 1972, Shooman proposed a software reliability mathematical model similar to the JM model. Musa gave his model in 1975. The time he discussed was divided into tests. The processor time required for the software execution and the software execution time for the official run after the test. After a tedious derivation, the relationship between the current MTTF and the execution time exponentially increases, and the relationship between the number of faults found and the execution time are given. In, Goel and Okumoto proposed an improved model of the J-M model, the Goel-Okumoto (G-O) model, which is also the most classic NHPP model. In 1983, Yamada and Osaki proposed a delayed S-shaped software reliability growth model. TE (testing-effort) appeared in the SRGM study in 1986, but it was not until the mid-1990s that the combination of the two was not deep. In the 21st century, various types of TE began to appear widely in SRGM. It becomes the norm to make the cost of integrating test resources into modeling. So far, hundreds of software reliability growth models have been proposed.

CONCLUSION

Software reliability is a measuring technique for defects that causes software failures in which software behavior is different from the specified behavior in a defined environment with fixed time. In this paper, various software reliability models are reviewed. Above analytical models are primarily useful in estimating and monitoring and it is viewed as a measure of estimation of software reliability and to enhance the quality of software.

REFERENCE

1. Wu, Y. P., Q. P. Hu, M. Xie& S. H. Ng, 2007. Modeling and analysis of software fault detection and correction process by considering time dependency. IEEE Transaction on Reliability, Volume 56 (4), p.629–642.

2. Xia, G., Zeephongsekul, P. & Kumar, S., 1992. Optimal software release policies for models incorporating learning in testing. Asia Pacific Journal of Operational Research, Volume 9, pp. 221-234.

3. Xia, G., Zeephongsekul, P. & Kumar, S., 1993. Optimal software release policy with a learning factor for imperfect debugging. Microelectron. Reliab., p. 81–86.

4. Xie, M., 1991. Software Reliability Modelling. : World Scientific.

5. Xie, M., 2000. Software reliability models—Past, present and future. Recent Advances in Reliability Theory: Methodology, Practice, and Inference, p. 325–340.

6. Xie, M., Zhao, M. &Schneidewind, T., 1992. Software Reliability Model Revisited. , Conference Proceedings, IEEE Computer Society Press, pp. 184-192.

7. Yamada, S., 2000. Software reliability models-past, present and future. Recent Advances in Reliability Theory: Methodology, Practice and Inference, pp. 323-340.

8. Yamada, S., Hishitani, J. & Osaki, S., 1991. Test-effort dependent software reliability measurement. International Journal of Systems and software, pp. 73-83.

9. Yamada, S., Hishitani, J. & Osaki, S., 1993. Software reliability growth model with Weibull Testing effort: a model and application. IEEE Transactions on Reliability, pp. 100-105.

10. Yamada, S., Ohba, M. & Osaki, S., 1983a. S-shaped reliability growth modeling for software error detection. IEEE Transactions on Reliability, p. 475–484.

11. Yamada, S. &Ohtera, H., 1990. Software reliability growth models for testing effort control. Eur. J. Oper. Res., p. 343–349.

12. Yamada, S., Ohtera, H. &Narihisa, H., 1986. Software reliability growth models with testing-effort. IEEE Trans. Reliab., Volume R-35, (1), p. 19–23.

13. Yamada, S. & Osaki, S., 1985a. Discrete Software reliability growth models. Applied Stochastic models and data analysis, Volume 1, pp. 65-77.

14. Yamada, S. & Osaki, S., 1985b. Software reliability growth modeling: Models and applications. IEEE Trans. Software Eng., Volume SE-11, p. 1431–1437.

15. Yamada, S., Tokuno, K. & Osaki, S., 1992a. Imperfect debugging models with fault introduction rate for software reliability assessment. Int'l J. System Science, vol 23(no. 12).

16. Yang, B. &Xie, M., 2000. A study of operational and testing reliability in software reliability analysis. ,Reliability Engineering and System Safety, Volume vol. 70, p. 323–329.

17. Yang, M. C. K. & Chao, A., 1995. Reliability-Estimation & Stopping-Rules for Software Testing, Based on Repeated Appearances of Bugs. IEEE TRANSACTIONS ON RELIABILITY, VOL. 44(NO. 2).

18. Zeephongsekul, P., Xia, C. & Kumar, S., 1994. A software reliability growth model primary errors generating secondary errors under imperfect debugging. IEEE Trans. Reliab., Volume R-43, (3), p. 408–413.

19. Zhang, N., Cui, G. & Liu, H., 2010. A Stochastic Software Reliability Growth Model with Learning and Change-point. Yichang, China, Proceedings of 2010 Conference on Dependable Computing (CDC'2010).

20. Goel, A. L. &Okumoto, K., 1979. Time-dependent error-detection rate model for software and other performance measures. IEEE Transactions on Reliability, Volume vol. 28, p. 206–211

Corresponding Author

Sharad Kumar Dubey*

Research Scholar, Shri Krishna University, Chhatarpur M.P.