ISSN:0975-9646

Overview: Software Reliability Growth Models

Ms. Shailee Lohmor#, Dr. B B Sagar*

#Research Scholar, Bharathiar University, Coimbatore, India

**Assistant Professor, Birla Institute of Technology, Mesra- Ranchi, India*

***Abstract*— An important attribute in software quality is software reliability. Before software delivered in to market it is thoroughly checked and errors are removed. Every software industry wants to develop software that should be error free. Reliability is the ability of the program to perform its required functions, whereas availability is the degree to which a system is operational and accessible when required for use. To the user this means that a system is more reliable if it correctly performs the tasks requested of it. A system is more available if you can use in anytime you want. Over past thirty years, many mathematical models have been proposed for estimation of reliability growth of product during software development process. Such models often referred as Software Reliability Growth Models (SRGM). Multiple models for measuring the reliability of the software and thus analysts are in a big chaos to decide which model should be used and which one is best. Thus, this review work depicts the overview and application of the SRGMs.**

***Keywords*— Software reliability, software reliability growth model, Residual Errors, Reliability Factor, Time Between Failure, Fault Count Model, Error Seeding Model, Input Domain Model**

I. INTRODUCTION

Reliability of software is possibility of no failure during a given operating time in a specified environment. Software reliability can be defined as the probability of failure-free software operation for a specified period of time in a specified environment [1],[2],[3],[4]. Software reliability growth models are helping the software industries to develop software which is error free and reliable. They try to predict software reliability from test data. These models try to show a relationship between fault detection data (i.e. test data) and known mathematical functions such as logarithmic or exponential functions. Software reliability growth models (SRGM) captures failure behaviour of software during testing and extrapolates it to determine its behaviour during operation. Hence this category of models uses failure data information and trends observed in the failure data to derive reliability prediction. The SRGM techniques are specifically useful for developers and testers during testing and debugging phase. This study aims to apply and compare the predictive capability of SRGM.

II. SOFTWARE RELIABILITY GROWTH MODELS

The software reliability growth is one of the fundamental techniques to assess software reliability quantitatively [1]. SRGM takes failure specification as the input and provides the reliability of the software as output [5]. The specifications used must be the number of failures within an interval and the time between two successive failures. The factors on which the failures depend on are the failure identification, removal and operational usage. The models applicable to the assessment of software reliability are called SRGM. SRGM are useful for estimating how software reliability improves as faults are detected and repaired. Software Reliability Models can be classified in two ways, one is based on Failure History and the other one is Data Requirements as depicted in the Fig 1:

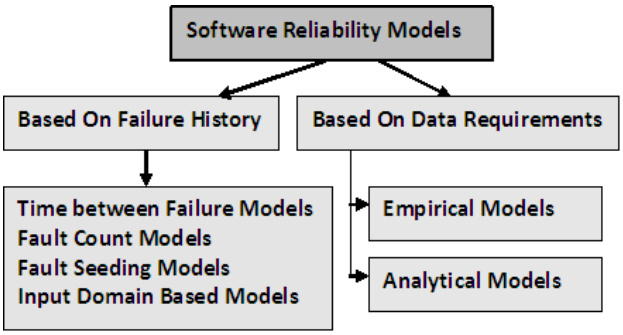


Fig. 1 Classification of Software Reliability Growth Models

The focus of this review study is on the software reliability model classification based on the failure history. The basic data requirement under each model category is summarized in Fig 2:

Model Category	Data Requirements
Time Between Failure	Time at each failure and failures are recorded
	Operational Profile and corresponding tests
Failure Count	Failure count in the interval and length of interval
Fault Seeding	Number of seeded faults
Category	Types and number of fault discovered

Fig. 2 Data requirements of the SRGM Models

1) Time between Failure Models

Under these models the study is based on the time between failures. It works on the assumption that the time between (i-1) th and ith failures is a random variable, which follows a distribution whose parameters depend on the number of faults remaining in the program during this interval. Estimates of the parameters are obtained from the observed values of time between failures, mean time to next failure, etc., are then obtained from the fitted model.

Jelinski Moranda Model

Jelinski Moranda (JM) model is an exponential model but is differs from geometric model in that the parameter used is proportional to the remaining number of faults rather than constant [6]. In JM model, we have N software faults at the start of testing, each is independent of others and is equally likely to cause a failure during testing. Fault removal technique is applied to remove defects and no new defects are introduced during debugging.

The basic assumptions of this model are:

1. There are a constant number of lines of code.
2. The operational profile of software is consistent.
3. Every fault has the same chance to be encountered during software operation.
4. Fault detection rate remains constant over intervals between fault occurrences.
5. Fault detection rate is proportional to current fault content of software.
6. Each detected error is corrected without delay.
7. Failures are independent

MODEL FORM:

The number of predicted errors or the mean value function, μ (tj), is given by

$$\mu (tj) =1/ b (a – (j-1))$$

Where b is the roundness or shape factor (the rate at which the failure rate decreases). a is the total number of software errors, and tj is occurrence time of jth fault . The number of Residual errors can be found out if the entire number of bugs is detected and is calculated as:

$$ER=a- \mu (tj)$$

The Reliability Factor is the measure of software reliability. Its values vary between 0 and 1. If RF=1, then software under consideration is perfect, however if RF=0, then the software is highly vulnerable. When RF approaches close to1 then the software can be considered as reliable.

$$RF=1-(ER/a)$$

2. Failure Count Models

The group refers to the models that are based on the number of failures that occur in each time interval. The random variable of interest is the number of faults (failures) occurring during specified time intervals. It is assumed that failure counts follow a known stochastic process. Usually a Poisson distribution with a time dependent will be discrete or continuous failure rate. The time can be calendar time or CPU time Parameters of the failure rate can be estimated from the observed values of failure counts and then the Software reliability parameters are obtained from the appropriate expression.

Goel- Okumuto Non- homogeneous Poisson Process Model

In this model Goel-Okumoto [9] assumed that a software system is subject to failure at random times caused by faults present in the system. The Non Homogeneous Poisson Process (NHPP) model is a Poisson type model that takes the number of faults per unit of time as independent Poisson random variables. The basic assumptions of this model are:

1. Cumulative number of failures by time t follows a Poisson process.
2. Number of faults detected in each time interval is independent for any finite collection of time intervals.
3. Defects are repaired immediately when they are discovered.
4. Defect repair is perfect. That is, no new defect is introduced during test.
5. No new code is added to software during test.
6. Each unit of execution time during test is equally likely to find a defect if the same code is executed at the same time.

MODEL FORM:

The mean value function or the cumulative failure counts must be of the form

$$\mu(t)=a(1- e^{-bt})$$

for some constants b>0 and N>0. a is the expected total number of faults to be eventually detected. In this model a is the expected number of failures to be observed eventually and b is the fault detection rate per fault.

3. Error or Fault Seeding Model

In the model of Error Seeding, a predefined number of artificially generated errors are "incorporated" in the program code. After that, test runs are used to detect the errors and to examine the ratio between actual and artificial errors based on the total number of detected errors. Naturally, the artificially generated errors are not known to the testers. In a first approach, the number of undetected errors can be estimated as follows:

$$FU = FG \cdot (FE / FEG)$$

Where FU refers to number of undetected errors, FG means number of not seeded errors detected, FE refers number of seeded errors and FEG as number of seeded errors detected. By seeding errors to a document and then let the document undergo testing of some kind it is possible to calculate how many real errors that exist. According to these, an estimation of the fault content of the program preceding to seeding is obtained and used to assess software reliability and other relevant measures. The basic assumptions of this model are:

1. Seeded faults are randomly distributed in the program.
2. Indigenous and seeded faults have equal probabilities of being detected.

Mills Hyper geometric model

Mills Hyper geometric model is one the model of the type fault seeding [7]. This model is based on approach that number of known faults be randomly seeded in the program to be tested. The program is then tested for some interval of time. Original indigenous fault count can be evaluated from the numbers of indigenous and seeded faults uncovered during the test by using the hyper geometric distribution.

4. Input - Domain Based Category

Input - domain based category includes models that assess the reliability of a program when the test cases are sampled randomly from well - known operational distribution of inputs program. By finding all unique paths through the program and then execute each and everyone it is possible to guarantee that everything is tested. Nelson model is the example of the type input domain [8]. The basic assumptions of this model are as follows:

1. Input profile distribution is known.
2. Random testing is used.
3. Input domain can be partitioned into equivalent classes.

Nelson model

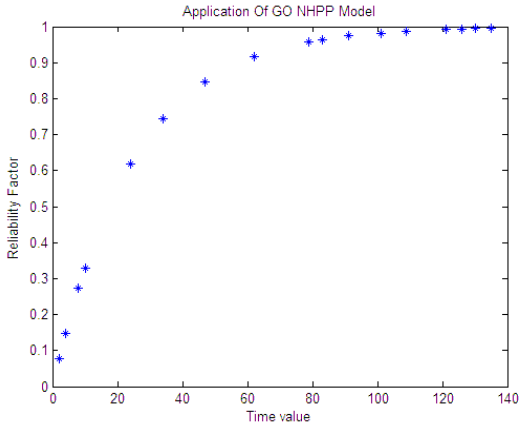
In this model, reliability of software is calculated by executing the software for a sample of n input. Inputs are randomly selected for the input domain set S= (Si, i=1,.....,N) and each Si is set of data values required for the execution. Probability distribution Pi I ; the set (Pi, i = 1, N) is the operational profile or simply the user input distribution. And random sampling is done according to this probability distribution. Suppose n_e is the number of execution that leads the execution to fail. Then estimation of reliability R1 is: $R1=\{1-n_e/n\}$.

ANALYSIS

In order to analyze the applicability of the models The data set used consists of 10 observations corresponding to times between testing. The total expected error (EE) in the code are 100. The roundness factor or the defect reduction rate is considered to be between 0.03 and 0.05 (based on empirical studies of several software's). The value of the roundness factor b depends upon the type of software and the environment in which it is being used. The term MVF refers to the mean value factor and RF to the reliability factor.

GO-NHPP Model		JM Model	
MVF	RF	MVF	RF
7.6884	0.0769	0.25	0.0025
14.7856	0.1479	0.2525	0.0025
27.3851	0.2739	0.2551	0.0026
32.968	0.3297	0.2577	0.0026
61.7107	0.6171	0.2604	0.0026
74.3339	0.7433	0.2632	0.0026
84.741	0.8474	0.266	0.0027
91.6257	0.9163	0.2688	0.0027
95.7574	0.9576	0.2717	0.0027
96.3847	0.9638	0.2747	0.0027
97.3748	0.9737	0.2778	0.0028
98.2403	0.9824	0.2809	0.0028
98.7222	0.9872	0.2841	0.0028
99.2093	0.9921	0.2874	0.0029
99.3526	0.9935	0.2907	0.0029
99.4483	0.9945	0.2941	0.0029
99.5483	0.9955	0.2976	0.003

Table 1 Analysis results of Jelinski – Moranda and Goel- Okumuto NHPP Model



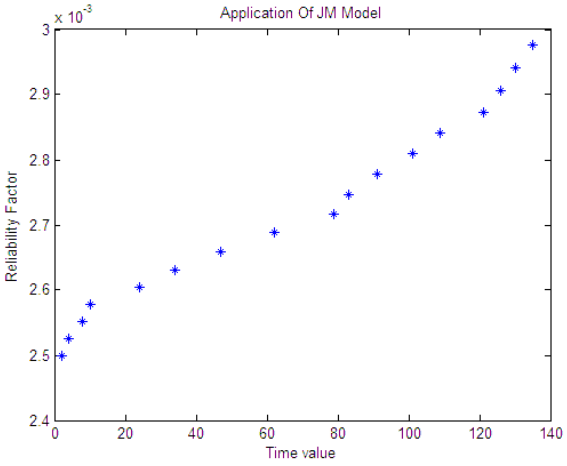


Fig 3: Plot of Reliability factor in JM and GO-NHPP Model

III. CONCLUSIONS

In this review work, effort made to provide an overview of some existing Software reliability models with their underlying assumptions. Further an experiment was made to analyze the applicability of the models via an example. On the basis of the assessment made by the data set used it has been observed that as the number of residual errors decreases the reliability factor increases and more the reliability factor is close to 1, it is said to be highly reliable. The assumptions made by the time between failure models are hard to meet as compared to the fault counting models.

REFERENCES

[1] Michael R. Lyu, Handbook of Software Reliability Engineering, IEEE Computer Society Press and McGraw-Hill Book Company , 2005.

[2] Musa J.D., Software Reliability Engineering: More Reliable Software, Faster Development and Testing, McGraw-Hill, 1999.

[3] J.D. Musa, A. Iannino, and K. Okumoto, Software Reliability:Measurement, Prediction, Application, McGraw-Hill Publishing Company, New York, NY, 1987

[4] Kapur P.K and Garg, " Cost reliability optimum release policies for a software system with testing effort", OR , Vol. 27, no 2 , pp 109-116,1990.

[5] Dr. William H. Farr , "A survey of Reliability Modeling and Estimation", Sept1983.

[6] Quadri, S.M.K, Mohd. Razeef, Nesar Ahmad, " A Comparative Overview of Software Reliability Growth Models" in IJARCS, vol2, No.1,pp 99-105, Jan-Feb 2011

[7] H. D. Mills, "On the statistical validation of computer programs, "IBM Federal Syst. Div., Gaithersburg, MD, Rep. 72-6015, 1972

[8] Razeef Mohd, Mohsin Nazir, " Software Reliability Growth Models: Overview and Applications", Journal of Emerging Trends in Computing and Information Sciences VOL. 3, NO. 9, SEP 2012

[9] A L Goel, "A time dependent error detection

[10] rate model for software reliability and other performance measures," IEEE Trans. Rel, Vol R-28, pp.206-211, 1979