International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 25307-25315

http://www.ripublication.com

Performance Analysis of Truncated Software Reliability Growth Models

Dr.Sangeetha Yalamanchili
Assistant Prof. Dept. of Information Technology
Velagapudi Ramakrishna Siddhartha Engineering College
Kanuru, Vijayawada, India
Sangeetha18.yalamanchili@gmail.com
Dr.Satya Prasad
Associate Prof. Computer Sciences and Technology
Acharya Nagarjuna University
Guntur, India.
profrsp@gmail.com

Abstract

Software Reliability plays an important role in software developments due to large scale applications of system software's. Software Reliability is the probability of failure free software operation for a specified period of time in a specified environment. It is the ability of a computer program to perform its intended functions and operations in a system's environment, without experiencing failure (system crash). In this paper, a few software reliability growth models (SRGM) are proposed. Such as GOEL model, Inflection S-shaped model and others.An analysis was performed comparing the right time of truncated growth models to present a better working model. The intensity function, mean-value function, reliability of the software, estimation of parameters and the special applications of these models are discussed using real data.

Keywords: Software reliability, SRGM, truncated, mean-value function, estimation of parameters

Introduction

In recent years, many developed s/w reliability growth model are based on the Non Homogenous Poisson Process [1]. All these developed models are based on the infinite testing time and therefore, it is the time consuming process. Software Reliability Growth Models with finite testing time is proposed. The truncated exponential SRGM [2] is developed using Non Homogeneous Poisson Process.

Quality is the desirable characteristic of any product. The growth of any company depends on the quality of products it produces. The quality of software can be judged by its correctness, adaptability and testability [3]. A software product has a typical development process. Some activities of the process are related to particular phases of development. The other activities of the software process called umbrella activities are applied throughout the software development. One of those activities is assessing quality of the software. As in the case of any other product to obtain quality software the only way is to identify the opportunities of improvement at every stage of software development. The tests and reviews conducted throughout the development come under the quality control.

Software reliability is the commonly used quality metric. This is estimated by the use of an analytical model whose unknown parameters are estimated from the available failure data. There are a number of views to quantify software reliability [8]. It may be binary in nature. A perfect program is said to be reliable while an imperfect one is said to be unreliable. The probability that software faults do not cause a failure during a specified exposure period in a specified environment is called software reliability. This nature is due to the uncertainty in the usage of the various software functions and the specified exposure period.

A process which develops in time in accordance with some probabilistic laws is called a stochastic process. They are used for the description of a systems operation over time. The two main types of stochastic processes are continuous and discrete. Counting processes in reliability engineering are widely used to describe the appearance of events in time, e.g. failures, number of perfect repairs, etc. The simplest counting process is a Poisson process. Poisson type models assume that the number of failures detected within distinct time intervals is independent.

1. Homogeneous Poisson Process (HPP): with the same rate of failure.
2. Non-Homogeneous Poisson process (NHPP): with a varying rate of failure.

Non-Homogenous Poisson Process

The NHPP group of models provides an analytical framework for describing the software failure phenomenon during testing. These models are usually based upon various debugging scenarios, and can catch quantitatively typical reliability growth observed in the testing phase of software products. The NHPP based SRGMs are proved to be quite successful in practical software reliability engineering [5] [6]. Many of the SRGMs assume that each time a failure occurs, the fault that caused it can be immediately removed and new faults are not introduced. It is usually called perfect debugging. Imperfect debugging models have proposed a relaxation of the above assumption (Pham, 1993).

If ‘t’ is a continuous random variable with probability density function (PDF): f $(\theta_1, \theta_2, \ldots, \theta_k)$ and cumulative distribution function (CDF): F(t) . Where, $\theta_1, \theta_2 \ldots \theta_k$ are k unknown constant parameters. The mathematical relationship between the PDF and CDF is given as: $f(t) = F'(t)$.

Let N (t) be the cumulative number of software failures by time ‘t’. A nonnegative Integer valued stochastic process N(t) is called a counting process, if N(t) represents

[0, t] the total number of occurrences of an event in the time interval and satisfies these two properties:

1. If *t1 < t2* , then *N(t1) ≤ N(t2)*
2. If *t1 <t2*, then *N(t2)– N(t1)*is the number of occurrences of the event in the interval [*t1*, *t2*].

One of the most important counting processes is the Poisson process. A counting process, N (t), is said to be a Poisson process with intensity λ if

1. The initial condition is *N(0)* = 0.
2. The failure process, *N(t)*, has independent increments.
3. The number of failures in any time interval of length *s* has a Poisson distribution with mean λs , that is,

$$P\{N(t+s)-N(t)=n\}=e^{-\lambda s}(\lambda s)n/n! \quad (1)$$

Proposed Work

A. Goel Model

An S-shaped software reliability growth model (SRGM) based on a non-homogeneous Poisson process (NHPP) with two types of errors has been proposed. The errors have been classified depending upon their severity. We have estimated the model parameters and obtained the optimum release policies which minimize the cost subject to achieving a given level of reliability. Numerical results illustrating the applicability of the proposed model are also presented. In this project, i considered the error detection phenomena in the testing phase when modifications or improvements can be made to the software in the testing phase [7].

The general assumption in this model is that all defects during the development and correction process do not introduce any new defects. Hence this ideal process of perfect debugging allows the reliability to increase throughout the testing process. Exponential SRGM and proposed SRGM fall in this category. The exponential SRGM is given by Goel and Okumoto [2]. It has a mean value function

$$m(t)=a[1-e^{-bt}], a>0, b>0 \quad (2)$$

where,

a is the initial error content in the software.

b is the error detection rate per error at time t.

Truncated Exponential SRGM is the proposed SRGM. It has an instantaneous failure rate which is given as follows.

$$\lambda(t)=\frac{c}{d} \quad (3)$$

Where, $c=abe^{-bt}d=1-e^{-bT}$

The graphical representation of instantaneous failure rate is shown in the below. The mean value function of this model is briefly discussed in the truncated software reliability growth model [4] also m(0) = 0 and m(T) = a.

B. Inflation S-Shaped Model

Quality of software and its reliability can be achieved by either eliminating their causes or by improving the process followed for development of software and their operational procedures. Deploying of statistical process control will affect the software process, since this process itself requires organizational commitment over different functional limits [9].

Software Reliability is one of the most important aspects of software quality. Software Reliability Engineering (SRE) is the quantitative study of the operational behavior of software-based systems with respect to user requirements concerning reliability. According to ANSI, Software Reliability is defined as: the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is defined as a probabilistic function, and comes with the notion of time. Mathematically, reliability R(t) is the probability that a system will be successful in the interval from time '0' to time 't':

$$R(t) = P(T > t)t \geq 0 \tag{4}$$

Where, 'T' is a random variable which denotes the time-to-failure. Software reliability models serve to aid the software engineer by indicating the likelihood of system operation over a given time interval according to the stated specifications. Software reliability predictions can be used to judge the quality of a program. It is important to define the key terms in developing and describing software reliability models. The following definitions are used to distinguish between failures and faults throughout the thesis.

A **failure** is a departure from how software should behave during operation according to the requirements.

A **fault** is a defect in a program, that when executed causes failure(s).

The study of software reliability can be categorized into three parts: modeling, measurement and improvement. There are two general schemes to demonstrate whether the required software reliability has been achieved or not. They are formal verification and statistical testing. Various approaches to estimate software reliability by statistical testing is reviewed in this section.

We now discuss a stochastic model for a software error detection process based on NHPP in which the growth curve of the number of detected software errors for the observed failure data is S-shaped, called Delayed S-shaped model. The software error detection process is described by an S-shaped curve can be characterized as a learning process in which test team-members become familiar with the test environment, testing tools, or project requirements i.e., their test skills gradually improve. The delayed S-shaped model is based on fallowing assumptions:

1. All faults in a program are mutually independent from the failure detection point of view.
2. The probability of failure detection at any time is proportional to the current number of faults in software.
3. The proportionality of failure detection is constant.
4. The initial error content of the software is a random variable.

5. A software system is subject to failures at random times caused by errors present in the system.
6. The time between (i-1)th and ith failures depends on the time to the (i-1) th failure.
7. Each time a failure occurs, the error which caused it is immediately removed and no other errors are introduced.

Assume,

$$b(t) = \frac{b^2 t}{bt} + 1 \tag{5}$$

Where b is the error detection rate per error in the steady state. The mean value function is given by

$$m(t) = a[1 - (1 + bt)e^{-bt}] \tag{6}$$

This shows an S-shaped curve. This model is called the delayed S-shaped NHPP model for such an error detection process, in which the observed growth curve of cumulative number of detected errors is S- shaped. The corresponding failure intensity function is

$$\lambda(t) = ab^2 te^{-bt} \tag{7}$$

For Type-1 data, the estimate of parameters a and b using the MLE method can be obtained by solving the following equations simultaneously:

$$a = y_n/[1 - (1 + bt_n e^{-bt})] \tag{8}$$

$$\frac{2n}{b} = \sum_{i=1}^{n} S_i + \frac{nb{s_n}^2 e^{-bt_n}}{[1-(1+bS_n e^{-bs})]} \tag{9}$$

a=71.725 b=0.104

After thoroughly scanning the literature, I am motivated to study the following research problems related to software reliability.

1. A Process control mechanism based on the cumulative observations of Time domain failure data using mean value function of Rayleigh model.
2. A Process control mechanism based on the cumulative observations of Interval domain failure data using mean value function of Rayleigh model.
3. A Process control mechanism based on cumulative observations of time between every rth failure (i.erth order) data using mean value function of the Rayleigh model.
4. As my interest got inclined to the work of Stieber (1997) that deal with detection of unreliable software components using the theory of Wald's sequential probability ratio test (SPRT) procedure as a statistical quality control tool. It is about a homogenous Poisson process. As no further publications on lines of Stieber are traced out by us, we attempt to adopt the methodology of Strieber to an NHPP with a general mean value function and specialize it for our specific NHPP based Rayleigh Software Reliability Growth Model both in Time domain as well as in Interval domain.

Our attempts and findings in these directions are presented in the chapters that follow with an aim of proposing analytical techniques for quality software.

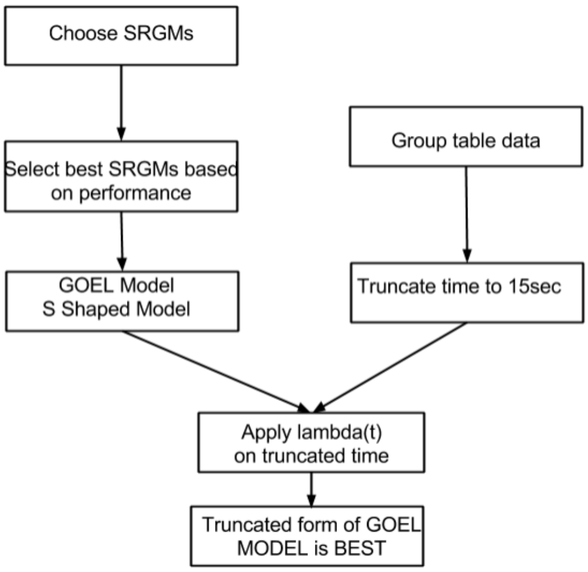
The inflated S shaped model [10] is based on the dependency of faults by postulating the following assumptions:

1. Some of the faults are not detectable before some other faults are removed.
2. The probability of the failure detection at any time is proportional to the current number of detectable faults in the software.
3. Failure rate of each detectable fault is constant and identical.
4. The isolated faults can be entirely removed.

Approch

Flow diagram is a graphic representation of the physical route or flow of people, materials, paperwork's, vehicles, or communication associated with a process, procedure plan, or investigation. In the second definition the meaning is limited to the representation of the physical route or flow. An example of such a diagram is the illustration of the flows in a nuclear submarine propulsion system, which shows different streams back and forth in the system. The representation of such a system can be supplemented by one or more flowcharts, which show the sequence of one of the flows in one direction, or any of the control flows to manage the system.

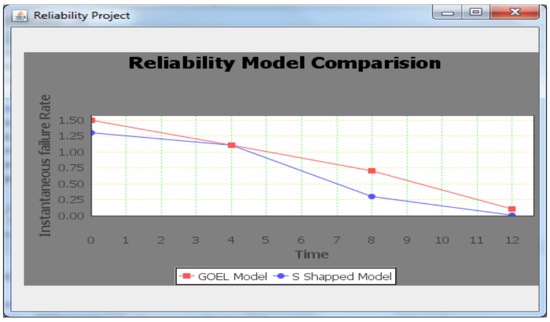
The physical movement of objects from one location to another can also be visualized in a mix of maps and flowchart or snaky diagram, which is called flow maps. Flow maps can indicate on a map, what flows, moves or migrates, in which direction, and in which quantities etc.



Conclusion

On working with different datasets provided by IBM we found that, this approach of finding fault is pretty much faster and provided with almost exact results compared with usual approach of finding faults.

Instantaneous Failure Rate for GOEL and S Shaped Models



In the above figure, We observe the Instantaneous Failure Rate for GOEL and S-Shaped Models. Here the x-axis indicated the time and y-axis indicated the instantaneous failure rate. Here the red line indicates the GOEL model and another line indicates the S-Shaped model. Here we clearly observed the two models by instantaneous failure rate and time. Fast result when compared to the previous infinite model. this graph results in better performance GOEL when compared to s-shaped.

Instantaneous Failure Rate for GOEL and S Shaped Models

In the given figure, we observe the Instantaneous Failure Rate for GOEL and S-Shaped Models. Here the x-axis indicated the time and y-axis indicated the instantaneous failure rate. Here the red line indicates the GOEL model and another line indicates the S-Shaped model. Here we clearly observed the two models by instantaneous failure rate and time. Fast result when compared to the previous infinite model. this graph results in better performance GOEL when compared to s-shaped

Instantaneous Failure Rate for GOEL and S Shaped Models

Comparison graph

In the given figure we show the comparison of three databases. In this we take datasets in x-axis and time is taken in y-axis. Here the red color indicates the S-Shaped model and another color indicates the GOEL model

Table 1: Comparitive Study

Model	*Sum of Weights*	*Model Weights*	*Model Values*
GOEL Model	1.075671	1.070750	1
Inflation S-Shapped Model	1.088970	1.122031	2
P-N-Z Model	1.137328	1.473892	3
Musa Okumoto Model	1.183498	1.626698	4

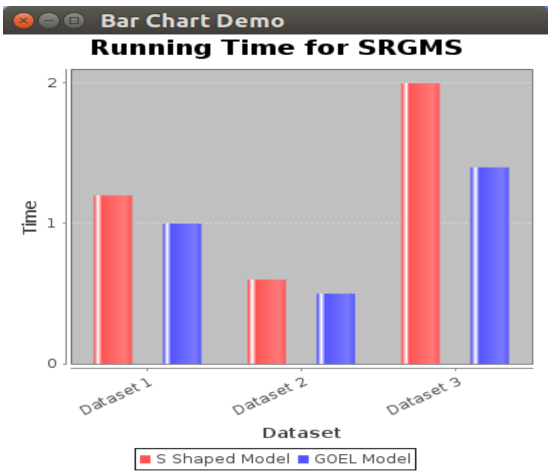


Figure 1: Comparitive Bar Graph

Concluding that Goel-Okumoto gives more accurate results and fast outputs when executed with instantaneous failure rates. Working with various datasets and comparing their results.

References

[1] D.R. Prince Williams, P. Vivekanandan, Truncated Software Reliability Growth Model,Korean J. Comput. & Appl. Math. Vol. 9.

[2] A. L. Goel, K. Okumoto, ”Time-dependent error-detection rate model for software reliability and other performance measures”. IEEE Trans. Reliab. 1979, R-28, pp. 206- 211.

[3] L.Y. Chan, M. Xie, T.N. Goh., “Cumulative quality control charts for monitoring production processes. Int J Prod Res 2000; 38(2):397-408.

[4] M. Kimura, S. Yamada, S. Osaki, ”Statistical Software reliability prediction and its applicability based on mean time between failures”. Mathematical and Computer Modelling,1995, Volume 22, Issues 10-12, Pages 149-155.

[5] A.K. Sheikh, J.K. Boah, M. Younas, Truncated Extreme value model for pipeline Reliability, Reliability Engineering and system safety, 25(1989), 1-14.

[6] J. Musa, K. Okumoto, Software Reliability: Measurement, Predication, Application,McGraw-Hill, Inc., New York, 1987.

[7] N. Kareer, P.K. Kapur, P.S. Grover, An S-Shaped Software Reliability Growth Models with Two types of Errors, Micro. Reliab. 30(6) (1990), 1085-1090.

[8] Obha, Software Reliability Analysis Models, IBM. J. Res. Deve. 28(4) (1984), 42-47

[9] S. Yamada, J. Hishitani, S. Osaka, Software Reliability Measurement and Assessment based on Nonhomogenous Poisson Process Models: A Survey, Micro. Reliab. 32(12) (1992), 1763-1773.

[10] Kimura, M., Yamada, S., Osaki, S., 1995. ”Statistical Software reliability prediction and its applicability based on mean time between failures”. Mathematical and Computer Modeling Volume 22, Issues 10-12, Pages 149-155.