Quest Journals
Journal of Software Engineering and Simulation
Volume 2 ~ Issue 2 (2014) pp: 01-11
ISSN(Online) :2321-3795 ISSN (Print):2321-3809
www.questjournals.org

Research Paper

Two Dimensional Software Reliability Growth Models Using Cobb-Douglas Production Function and Yamada S-Shaped Model

B.Anniprincy[1] & Dr. S. Sridhar[2]
[1]*Research Scholar,Sathyabama University,Chennai,*
[2]*Dean-Cognitive & Central Computing Facility, RV College of Engineering, Bangalore-560059*

Received 10 February, 2014; **A**ccepted 21 February, 2014 © The author(s) 2014. Published with open access at **www.questjournals.org**

ABSTRACT:- Software reliability is the likelihood of the failure free function of software in a given period of time under some certain circumstances. Software testing can be defined as the process to detect the faults in totality and worth of developed computer software. Testing is very much important in assuring the quality of the software by identifying faults in software, and also most possibly removing them to make the software more efficient. But testing of the software for a long time may not ensure a bug free software and high reliability. Optimum amount of code also needs to be covered to make sure that the software is of good quality. Testing time alone may not give the correct picture of the number of faults removed in the software. Therefore to capture the combined effect of testing time and testing coverage we propose two dimensional software reliability growth models by using Cobb-Douglas production function by incorporating the effect of testing time and testing coverage on the number of faults removed in the software system. In this paper, we develop an S-shaped model with imperfect debugging and fault generation to solve the above issues occurred during the testing of software. The proposed model is validated on real data sets.

Keywords:- *Software Reliability, Two dimensional, Non-Homogeneous Poisson Process (NHPP), Testing Coverage (TC), Cobb-Douglas Model, Imperfect Debugging, S-Shaped Model.*

I. INTRODUCTION

The function of software is intensifying quickly in recent society. Hence, quality, reliability, and customer satisfaction become the main goals for software engineers and very important considerations for software development organizations. Testing is the major quality control used during software development [7]. The quality of the software system has many attributes such as maintainability, portability, usability, security, reliability, availability, etc. Software reliability is the most dynamic attribute which can measure and predict the operational quality of the product [3]. The aim of software reliability engineers is to increase the probability that a designed program will work as intended in the hands of the customers [1]. Modeling of software reliability accurately and predicting its possible trends are essential for determining overall reliability of the software. It is easy to determine some important metrics like time period, number of remaining faults, mean time between failures (MTBF), and mean time to failure (MTTF) through SRGMs [6].

Research has been conducted in software reliability engineering over past three decades, and many software reliability growth models (SRGM) have been proposed [2]. The stochastic characteristics for the software failure occurrence or the software fault detection phenomenon changes due to changing the fault-target, the difference of the fault density for each module, and so forth [5]. In the past years, several SRGMs based on NHPP which incorporates the testing effort functions (TEF) have been proposed by many authors (Yamada et al., 1984; 1986; 1993; Yamada and Ohtera, 1990; Huang et al., 2007; Kuo et al., 2001; Bokhari and Ahmad, 2006; Quadri et al., 2006). Recently, Bokhari and Ahmad (2007) and Ahmad et al. (2008; 2010) also proposed a new SRGM with the exponentiated Weibull (EW) testing-effort functions to predict the behavior of failure and fault of software [4]. Software is one of the important safety issues in digital system safety assessment. When using safety-critical software, various methods like formal verification and validation play critical roles in demonstrating compliance with several regulatory requirements [8].

This mathematical model enables us to describe a software reliability growth process observed in the actual testing-phase by treating the software failure-occurrence or the soft-ware fault-detection phenomenon as random variables [5]. In the context of software testing, the key elements are the testing effort, and effectiveness of the test-cases. Many published models either assumes that the consumption rate of testing resources is constant, or do not explicitly consider neither the testing effort nor its effectiveness [1]. During software testing phase, much testing-effort is consumed itself. The consumed testing-effort indicates how the errors are detected effectively in the software and can be modified by different distributions [6]. One of the most serious limitations is the expected total number of inherent software faults calculated by the software reliability growth models that are highly sensitive to time-to-failure data [8]. However, there are several limitations when applying the software reliability growth models to safety-critical software. One of the most serious limitations is that the expected total number of inherent software faults calculated by the software reliability growth models is highly sensitive to the time-to-failure data [14].

When software testing results or software failure histories are available, we can use a process of directly estimating the reliability of a piece of software by using various software reliability growth models, such as Musa Basic Model, Goel Okumoto Non-Homogeneous Poisson Process (NHPP) Model, Musa Okumoto NHPP Model, Jelinski Moranda Model, Musa Poisson Execution Time Mode, Littlewood Verall Model, Weibull Model, Raleigh Model, Exponential Model, Logarithmic Poisson Model, Delayed S-shaped Growth Model, Imperfect Debugging Model, Inflection S-shaped Growth Model, and Logistic Model. Each model has its own best fitting failure data. This means that the model selection is very crucial to producing meaningful reliability predictions [8]. Within these models one can distinguish two main categories: predictive models, assessment models. Predictive models typically address the reliability of the software early in the life cycle at the requirements or at the preliminary design level or even at the detailed design level in a waterfall life cycle process or in the first spiral of a spiral software development process. Predictive models could be used to assess the risk of developing software under a given set of requirements and for specified personnel before the project truly starts. Assessment models evaluate present and project future software reliability from failure data gathered when the integration of the software starts.

An important feature of many systems is growth or change of some of their characteristics over time, which has to be taken into account when constructing a statistical model for the system. For example, a common approach for measuring software reliability is by using a statistical model whose parameters are generally estimated from available data on software failures, and the model may be obtained by observing the overall trend of reliability growth during the debugging process. In other words, a software reliability growth model describes how observation of failures and correcting the underlying faults, such as occurs in software development when the software is being tested and debugged, affect the reliability of software [10]. To ensure analytical tractability, most of these models assume that a software fault is fixed immediately upon detection, and that no new faults are introduced during the debugging process. In practice, however, the time taken to debug a fault is finite, and this debugging time has a direct impact on the residual number of faults, and hence the reliability of the software application [16].

Software reliability is widely recognized as one of the most important aspects of software quality, and it spawns much research effort into developing methods of quantifying it. The ultimate goal when developing SRGM is development of good reliability inference and prediction methods which can be applied to software development [17]. The residual faults in the software system directly contribute to the failure rate, causing software unreliability. Therefore, the problem of measuring software reliability can be approached by obtaining the estimates of the residual number of faults in the software. The number of faults that remain in the code is also an important measure for the software developer, from the point of view of planning maintenance activities [18]. Studies have shown that most of the faults encountered by customers are the ones that are reintroduced during debugging of the faults detected during testing. Thus imperfect debugging also affects the residual number of faults in the software, and can at times be a major cause of its unreliability and hence customer dissatisfaction.

II. RELATED WORKS

Carina Andersson [7] presented a replication of a method for selecting software reliability growth models to decide whether to stop testing and release software. They have applied the selection method in an empirical study, conducted in a different development environment than the original study. The results of the replication study shown that with the changed values of stability and curve fit, the selection method works well on the empirical system test data available, i.e., the method was applicable in an environment that was different from the original one. The application of the SRGMs to failures during functional testing resulted in predictions with low relative error, thus providing a useful approach in giving good estimates of the total number of failures to expect during functional testing.

Over the last several decades, more number of Software Reliability Growth Models (SRGM) have been developed to greatly facilitate engineers and managers in tracking and measuring the growth of reliability as software is being improved. However, some research work indicates that the delayed S-shaped model may not fit the software failure data well when the testing-effort spent on fault detection is not a constant. Chin-Yu Huang *et al.* [1] have first reviewed the logistic testing-effort function that can be used to describe the amount of testing-effort spent on software testing. They described how to incorporate the logistic testing-effort function into both exponential type, and S-shaped software reliability models. The proposed models are also discussed under both ideal, and imperfect debugging conditions. Results from applying the proposed models to two real data sets are discussed, and compared with other traditional SRGM to show that the proposed models can give better predictions, and that the logistic testing-effort function is suitable for incorporating directly into both exponential-type, and S-shaped software reliability models.

Lev V. Utkin *et al.* [10] proposed a new framework which was explored for combining imprecise Bayesian methods with likelihood inference, and it is presented in the context of reliability growth models. The main idea of the framework is to divide a set of the model parameters of interest into two sub-sets related to fundamentally different aspects of the overall model, and to combine Walley's idea of imprecise Bayesian models related to one of the sub-sets of the model parameters with maximum likelihood estimation for the other subset. In accordance with the first subset and statistical data, the imprecise

N. Ahmad *et al.* [4] proposed a material which compares the predictive capability of two popular software reliability growth models (SRGM), say exponential growth and inflection S-shaped growth models. They first reviewed the exponentiated Weibull (EW) testing-effort functions and discussed the exponential type and inflection S-shaped type SRGM with EW testing-effort. Then they analyzed the actual data applications and compared the predictive capability of those two SRGM graphically. The findings reveal that inflection S-shaped type SRGM had better prediction capability as compared to exponential type SRGM.

P. K. Kapur *et al.* [2] proposed two general frameworks for deriving several software reliability growth models based on a non homogeneous Poisson process (NHPP) in the presence of imperfect debugging and error generation. The proposed models are initially formulated for the case when there is no differentiation between failure observation and fault removal testing processes, and then extended for the case when there is a clear differentiation between failure observation and fault removal testing processes.

S. M. K. Quadri *et al.* [6] proposed a method for constructing software reliability growth model based on Non-Homogeneous Poisson Process. In that method, they have considered the case where the time dependent behaviors of testing-effort expenditures are described by Generalized Exponential Distribution (GED). Software Reliability Growth Models (SRGM) based on the NHPP are developed which incorporates the (GED) testing-effort expenditure during the software-testing phase.

III. SOFTWARE RELIABILITY GROWTH MODELS

Software Reliability is the probability of failure free operation of software in a provided time period under specified conditions. Software testing is a process to detect faults in the totality and worth of developed computer software. Testing is very important tool in assuring the quality of the software by identifying different faults in software, and possibly removing them. But testing of this software for a long time may not ensure a bug free software and high reliability. Best possible amount of code also needs to be covered to make sure that the software is of good quality. Testing time alone may not give the correct preview of the number of faults removed in the software. Therefore to capture the combined effect of testing time and testing coverage we propose two dimensional software reliability growth models.

IV. SOFTWARE RELIABILITY GROWTH MODELS WITH TWO TYPES OF IMPERFECT DEBUGGING

In this paper we will develop a two-dimensional model which shows the united effect of testing time and testing coverage to remove the faults mendacious dormant in the software. We will assume that the number of faults detached in the software by a fixed time is dependent on the total testing resources accessible to the testing team. This testing resource will be a fusion of both testing time and testing coverage. We have used cobb-douglas production function to develop the two dimensional model incorporating the effect of testing time and testing coverage on the number of faults removed in the software system. The faults in the software may not be removed perfectly. When the faults are not removed perfectly and lead to further generation of faults. In this paper, we develop an s shaped model with imperfect debugging and fault generation. The proposed method is implemented using JAVA and it is validated on real data sets.

Time Dependent Model

The time dependent behavior of fault removal process is explained by a Software Reliability Growth Model (SRGM). Most of the software reliability models can be categorized under Non Homogeneous Poisson Process (NHPP) models. The assumption that governs these models is software failure occurs at random times during testing caused by faults lying dormant in software. And, for modeling the software fault detection phenomenon, counting process $\{N(t); t \geq 0\}$ is defined which represents the cumulative number of software faults detected by testing time t. The SRGM based on NHPP is formulated as:

$$\Pr\{N(t) = n\} = \frac{m(t)^n \cdot \exp(-m(t))}{n!} \tag{1}$$

Where

$n = 0, 1, 2, 3\ldots.$

$m(t)$ is the mean value function of the counting process $N(t)$

Testing Coverage Based Modeling

The testing coverage based software reliability growth model can be formulated as follows:

$$\frac{dm(t)}{dt} = \frac{c'(t)}{1-c(t)}\left(N - m(t)\right) \tag{2}$$

Where,

$m(t)$ is the expected number of faults identified in the time interval $(0,t]$

$c(t)$ is the testing coverage as a function of time t.

N is the constant, representing the number of faults lying dormant in the software at the beginning of testing.

Here $c(t)$ defines the percentage of the coded statements that has been observed till time t. So, $1-c(t)$ defines the percentage of the coded statements which has not yet been covered till time t. Then, the first order derivative of $c(t)$, denoted by $c'(t)$, represents the testing coverage rate.

Therefore, function $\frac{c'(t)}{1-c(t)}$ can be taken as a measure of the fault detection rate. In one dimensional SRGM with testing coverage we need to define coverage function $c(t)$ although in a two dimensional modeling approach we need not define a coverage function and it can be estimated directly from the data.

S-Shaped Flexible Model

In 1992 Kapur and Garg developed an S-shaped model with assuming that when we remove the different faults in the software some additional faults in the software are removed without actually affecting the system. The revised Kapur garg model is derived by using a logistic rate as the detection rate to capture the effect of imperfect debugging and fault generation. This model was based on the assumption of Non-Homogeneous Poisson Process. The basic assumptions of the model are as follows:

1. Failure /fault removal phenomenon is modeled by NHPP.
2. Software is subject to failures during execution caused by faults remaining in the software.
3. Failure rate is equally affected by all the faults remaining in the software.
4. Fault detection / removal rate may change at any time moment.

The differential equation of the representing the rate of change of cumulative number of faults detected in time t is given as Eq. (3)

$$m'(t) = \frac{b}{1-\beta\exp(-bt)}\left(N - m(t)\right) \tag{3}$$

The below Eq. (4) gives the mean value function of the number of faults detected in time t

$$m(\tau) = \frac{N(1-\exp(-b\tau))}{1+\beta\exp(-b\tau)} \tag{4}$$

Where,

b is the rate at which a fault is detected/removed in the software.
m is the mean number of faults detected/ Corrected corresponding to testing time t.
β is the constant.
x is the rate of error generation.
p is the probability of imperfect debugging.

Two-Dimensional Modeling

Later a two dimensional software reliability model was developed to access the software quantitatively. The need of development of a two dimensional model is one of the ideal solution to the problem regarding software reliability in the hands of software engineers. In one dimensional analysis the object variable usually depends upon one basic variable although the object takes on many different roles based upon its dependence on various other factors. Two dimensional models are used to capture the joint effect of testing time and testing coverage on the number of faults removed in the software. Traditionally used one dimensional model was depending upon the testing time, testing effort or testing coverage. However if the reliability of a software is measured on the basis on the number of hours spent while testing the software or the percentage of software that was covered then the results are not conclusive. To handle the need of high precision software reliability we have the requirement of a software reliability growth model which does not only solve the issues related to the testing time but also the testing coverage of the software i.e. the percentage of code covered of the software. For this we have developed a two dimensional software reliability growth model which takes into account the joint effect of testing time and testing effort on the number of faults removed in the software. The two dimensional model developed in this paper is based on the Cobb Douglas production function.

Cobb Douglas Production Function

The Cobb–Douglas functional form of production functions is broadly used to represent the relationship of an output to inputs. It was proposed by Knut Wicksell (1851–1926), and tested against statistical evidence by Charles Cobb and Paul Douglas in 1900–1928. The Cobb-Douglas function considered a simplified view of the economy in which production output is determined by the amount of labor involved and the amount of capital invested. Even if there are many factors affecting economic performance, still their model proved to be remarkably accurate.

The mathematical form of the production function is given as follows

$$Y = AL^{\nu}K^{1-\nu} \tag{5}$$

Where,

Y is the total production per year.
L is the labor input.
K is the capital input.
A is the total factor productivity.
ν is the elasticity of labor which is constant and determined by available technology.

Fig. 1 shows the total production influenced due to change in the proportion of labor and capital in graphical form.

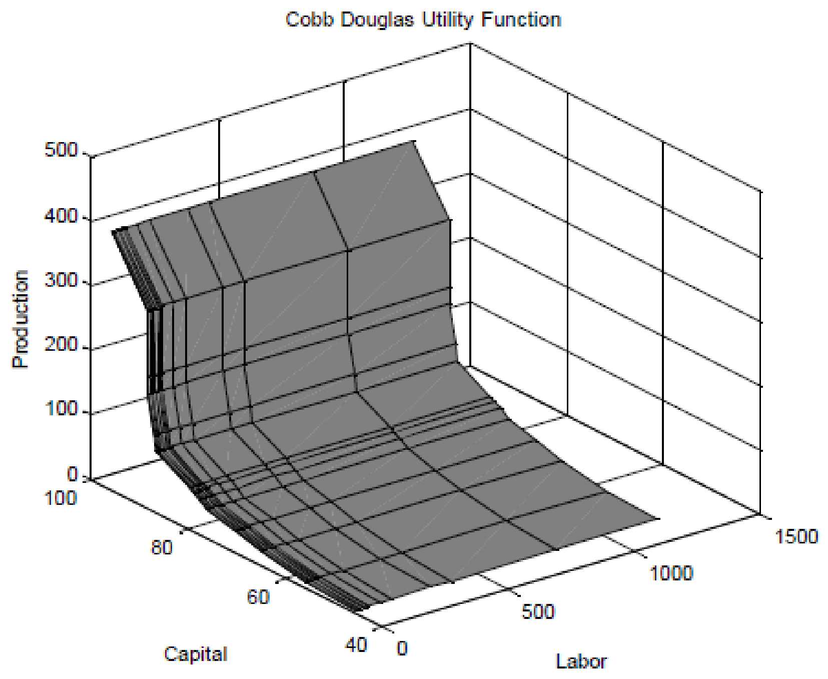

Fig. 1 Two-input Cobb-Douglas production function.

Cobb and Douglas made some assumptions which are stated as follows:
1. If either labor or capital vanishes, then so will production.
2. The marginal productivity of labor is proportional to the amount of production per unit of labor.
3. The marginal productivity of capital is proportional to the amount of production per unit of capital.

Cobb Douglas Model

Software Testing involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they shouldn't or things don't happen when they should. It is oriented to detection. The testing team has many resources of testing to make sure that software hence formed is of quality. These include software testing man hours, CPU time, testing effort testing coverage etc.

$$\tau \cong s^{\alpha} u^{1-\alpha} \qquad 0 \le \alpha \le 1 \tag{6}$$

Where,

τ is the testing resources
s is the testing time
u is the testing coverage
α is the Effect of testing time

Let $\{N(s,u), s \ge 0, u \ge 0\}$ be a two-dimensional stochastic process representing the cumulative number of software failures by time s and testing coverage u. A two-dimensional NHPP with a mean value function $m(s,u)$ is formulated as

$$\Pr(N(s,u)=n)=\frac{(m(s,u))n}{n!}\exp(-m(s,u)), n=0,1,2\ldots \tag{7}$$

$$m(s,u)=\int_0^s\int_0^u \lambda(\zeta,\xi)d\zeta d\xi \tag{8}$$

Two-Dimensional S-Shaped Model

In this proposed method, we develop a two dimension S-shaped model determining the combined effect of testing time and testing coverage. The differential equation of the representing the rate of change of cumulative number of faults detected with respect to the total testing resources is given as

$$m'(\tau) = \frac{b}{1+\beta \exp(-b\tau)}(N - m(\tau)) \tag{9}$$

The mean value function of the number of faults detected with testing resources x using the initial condition $x(0) = 0$ is given as

$$m(\tau) = \frac{N(1-\exp(-b\tau))}{1+\beta \exp(-b\tau)} \tag{10}$$

Now we extend the testing resource of one dimensional S-shaped model to a two dimensional problem. Using the cobb-douglas production the corresponding mean value function is given as

$$m(\tau) = \frac{N\left(1-\exp\left(-bs^{\alpha}u^{1-\alpha}\right)\right)}{1+\beta \exp\left(-bs^{\alpha}u^{1-\alpha}\right)} \tag{11}$$

In the above two-dimensional mean value function, if $\alpha = 1$, then the above mean value function can be regarded as a traditional one dimensional time dependent SRGM and if $\alpha = 0$ it becomes a testing coverage dependent SRGM.

Two-Dimensional S-Shaped Model with Imperfect Debugging

Mostly, the debugging process in real life won't be much perfect. While during the fault removal process two possibilities can occur. It may happen that the fault, which was considered to be perfectly fixed, had been improperly repaired and resulted in same type of failure again. There is also a fine chance that some new faults might get introduced during the course of correcting. This situation is much dangerous than the former one, because in the first case the total fault content is not altered, whereas in latter, error generation resulted in increased fault content. The effects of both type of imperfect debugging during testing phase are incorporated in our proposed model. The rate equation of flexible model with imperfect debugging and error generation can be written as follows

$$\frac{d}{dt}m(\tau) = \frac{bp}{1+\beta \exp^{-b\tau}}\left[N + xm(\tau) - m(\tau)\right] \tag{12}$$

We use logistic function to incorporate the effect of imperfect debugging and error generation. By solving the above equation using initial condition $N(0) = 0$, we get

$$N(t) = \frac{N}{1-x}\left\{1 - \left(\frac{(1+\beta)}{1+\beta \exp^{-bp\tau}}\right)^{p(1-x)} \exp^{-b\tau p(1-x)}\right\} \tag{13}$$

Reliability Evaluation

Software evaluation is a very significant phenomenon in quantitative software reliability assessment. The software reliability function signifies the probability that a software failure does not occur in time-interval $(t, t+x)(t \geq 0, x \geq 0)$given that the testing team or the user operation has been going up to time t. In two dimensional SRGM, we can assess software reliability in an operation phase where we assume that the testing coverage is not expanded. We can derive the probability that the software failure does not occur in time-interval $[s_\pi, s_\pi + \omega](s_\pi > 0, \omega > 0)$ that testing has been going up to s_π and the value of testing coverage has been attained up to u_π by testing termination time s_π as:

$$R(\omega / s_\pi, u_\pi) = \exp\{-[m((s_\pi + \omega), u_\pi / k) - m(s_\pi, u_\pi / k)]\} \tag{14}$$

Where k indicates the set of parameter estimates of a two dimension SRGM

V. RESULTS AND DISCUSSION

An SRGM is defined as a tool that can be used to evaluate the software quantitatively, develop test status, schedule status, and monitor the changes in reliability performance. Software reliability assessment and prediction is important to evaluate the performance of software system. In this paper, an effective software reliability growth model is developed with two types of imperfect debugging. In this section, the sample outputs are explained which is obtained during the execution of program.

Here the reliability of the software is identified by using S-shaped Cobb-Douglas function. In this paper, testing time and testing coverage was considered to identify the reliability of the software. A data set with failure number, failure interval and also day of failure is given as the input to the SRGM tool. The tool identifies the faults and gives the reliability parameters as output as shown in the figure below.

Dataset Result 1

FEATURES	VALUES
No of Faults in the Begining	358
The Mean value of the no of Failure	137.0
The Mean value of the Failure Interval	652.0735294117648
Cumulative no of Faults	2.7552232008292164E288
Mean value of no of Falut	3.936033144041738E287
The Result of Imperfect Debugging	1.7914796041770543E105
The Result of Software Reliablity is	8.266568798943481E8
Total Resource Process	1.0195310710476781E8
Total Production Process is	3.188107501070248E-185
TwoDSshapped Model Simulation	1.7914796041770543E105

Dataset Result 2

FEATURES	VALUES
No of Faults in the Begining	1618
The Mean value of the no of Failure	39.0
The Mean value of the Failure Interval	1772.6842105263158
Cumulative no of Faults	7.388594469044448E58
Mean value of no of Falut	1.0555134955777783E58
The Result of Imperfect Debugging	1.7914796041770543E105
The Result of Software Reliablity is	8.266568798943481E8
Total Resource Process	1.0195310710476781E8
Total Production Process is	3.188107501070248E-185
TwoDSshapped Model Simulation	1.7914796041770543E105

Fig. 2 Sample output of the SRGM Tool

Comparative Analysis

Using the proposed imperfect-debugging model, we now show a real numerical illustration for software reliability measurement. Here, in order to validate the imperfect-debugging model, the AE and MSF are selected as the evaluation criteria.

The Accuracy of Estimate (AE) is defined as

$$AE = \left| \frac{M_a - a}{M_a} \right| \tag{15}$$

Where M_a is the actual cumulative number of is detected errors after the test, and a is the estimated number of initial errors. For practical purposes, M_a is obtained from software error tracking after software testing.

The mean of Squared Errors (Long-term predictions) is defined as

$$MSE = \frac{1}{k}\sum_{i=1}^{k}\left[m(t_i) - m_i\right]^2 \tag{16}$$

Where $m(t_i)$ is the expected number of errors at time t_i estimated by a model, and m_i is the observed number of errors at time t_i. MSE gives the qualitative comparison for long-term predictions. A smaller MSE indicates a minimum fitting error and better performance.

The proposed method is compared with Yamanda Rayleigh Model and Huang Logistic Model. The comparison values of the proposed method, and Yamanda Rayleigh model and Huang Logistic Model are given in the below table.

Table I. Comparative results of different SRGM

Model	a	r	AE(%)	MSE
Proposed Model	628.87	0.0824	69.97	83.29
Yamanda Rayleigh Model	565.35	0.0196	57.91	122.09
Huang Logistic Model	394.08	0.0427	10.06	118.59

The graphical representation of AE and MSE for the proposed method, Yamanda Rayleigh model and Huang Logistic Model are shown in the below graphs

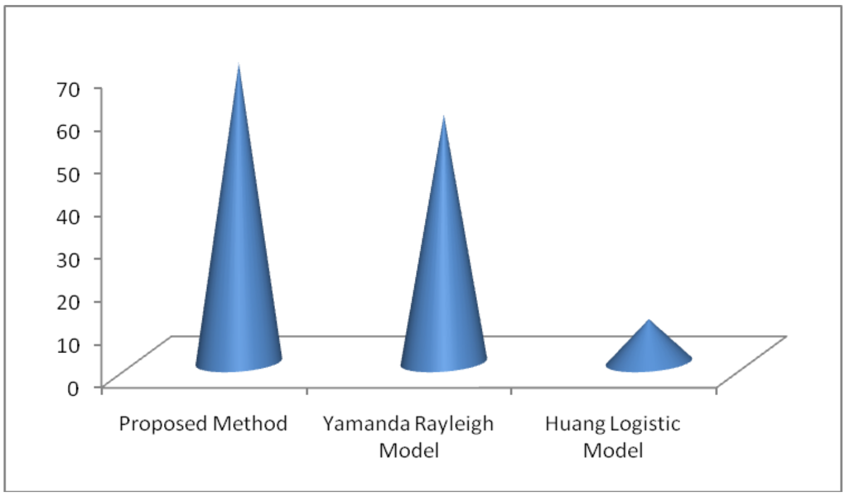

Fig.3 Comparision of AE

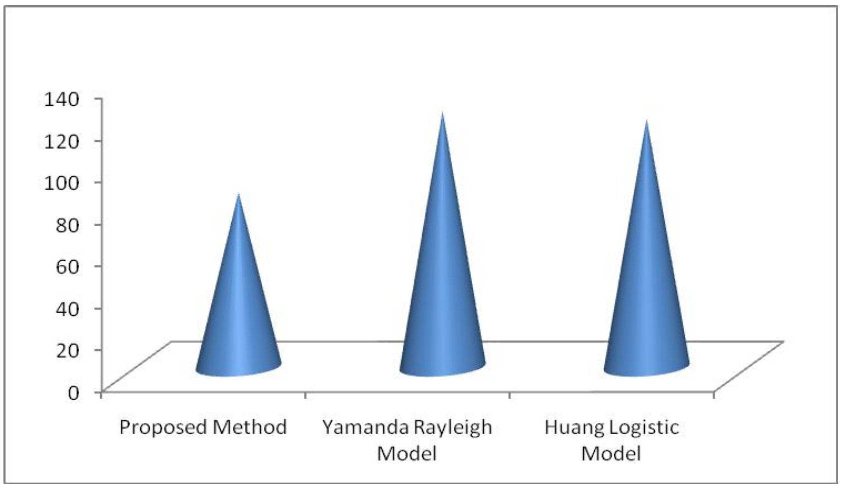

Fig.4 Comparision of MSE

From the above table and graphs, AE is very high and MSE is low than previous methods. Thus the proposed method is very effective.

VI. CONCLUSION

Software reliability engineering uses quantitative measurement to increase the efficiency of the testing effort. By developing operational profiles of the systems use, SRE requires that trade-offs between time, cost, and quality be made explicitly for the project. In this paper we have developed a general approach in deriving more general models based on simple assumptions, constant with the basic software reliability growth modeling based on NHPP. The proposed models implant a broader theoretical framework which accounts for interaction between different dimensions of software reliability metrics. Incorporating the dynamics of testing time of the software and the testing coverage has allowed us the model to be a two dimensional framework. The proposed models use the Cobb Douglas production function to capture the combined effect of testing time and testing coverage. The proposed models are validated on real data sets and analyses are done using goodness of fit criterion. We also conclude that the proposed SRGM has better performance as compare to the other SRGM and gives a reasonable predictive capability for the actual software failure data. Therefore, this model can be applied to a wide range of software.

REFERENCES

[1]. Chin-Yu Huang, Sy-Yen Kuo and Michael R. Lyu, "An Assessment of Testing-Effort Dependent Software Reliability Growth Models," IEEE Transactions on Reliability, Vol. 56, No. 2, pp. 198-211, Jun 2007.

[2]. P. K. Kapur, H. Pham, Sameer Anand and Kalpana Yadav, "A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation," IEEE Transactions on Reliability, Vol. 60, No. 1, pp. 331-340, Mar 2011.

[3]. Khurshid Ahmad Mir, "A Software Reliability Growth Model," Journal of Modern Mathematics and Statistics, Vol. 5, No. 1, pp. 13-16, 2011.

[4]. N. Ahmad, S. M. K Quadri and Razeef Mohd, "Comparison of Predictive Capability of Software Reliability Growth Models with Exponentiated Weibull Distribution," International Journal of Computer Applications, Vol. 15, No. 6, pp. 40-43, Feb 2011.

[5]. Shinji Inoue and Shigeru Yamada, "A Bivariate Software Reliability Model with Change-Point and Its Applications," American Journal of Operations Research, Vol. 1, No. 1, pp. 1-7, Mar 2011.

[6]. S. M. K. Quadri, N. Ahmad and Sheikh Umar Farooq, "Software Reliability Growth modeling with Generalized Exponential testing –effort and optimal Software Release policy," Global Journal of Computer Science and Technology, Vol. 11, No. 2, pp. 27-42, Feb 2011.

[7]. Carina Andersson, "A replicated empirical study of a selection method for software reliability growth models," Journal of Empirical Software Engineering, Vol. 12, No. 2, pp. 161–182, Apr 2007.

[8]. Han Seong Son, Hyun Gook Kang and Seung Cheol Chang, "Procedure for Application of Software Reliability Growth Models to NPP PSA," Journal of Nuclear Engineering and Technology, Vol. 41 No. 8,pp. 1065-1072, Oct 2009.

[9]. V. B. Singh1; Kalpana Yadav, Reecha Kapur and V. S. S. Yadavalli, "Considering the Fault Dependency Concept with Debugging Time Lag in Software Reliability Growth Modeling Using a Power Function of Testing Time," International Journal of Automation and Computing, Vol. 4, No. 4, pp. 359-368, Oct 2007.

[10]. Lev V. Utkin, Svetlana I. Zatenko and Frank P.A. Coolen, "Combining imprecise Bayesian and maximum likelihood estimation for reliability growth models," In Proc. of the Sixth International Symposium on Imprecise Probability: Theories and Applications, Durham, UK, 2009.

[11]. Dr. R. Satya Prasad, K. Ramchand H Rao and Dr. R.R.L. Kantha, "Software Reliability Measuring using Modified Maximum Likelihood Estimation and SPC," International Journal of Computer Applications, Vol. 21, No.7, pp. 1-5, May 2011.

[12]. Andy Ozment, "Software Security Growth Modeling: Examining Vulnerabilities with Reliability Growth Models," Journal of Advances in Information Security, Vol. 23, No. 2, pp. 25-36, 2006.

[13]. Martin Baumer, Patrick Seidler, Richard Torkar and Robert Feldt, "Predicting Fault Inflow in Highly Iterative Software Development Processes: An Industrial Evaluation," In Proc. of the 19th IEEE International Symposium on Software Reliability Engineering, Seattle, USA, 2008.

[14]. Man Cheol Kim, Seung Cheol Jang and Jae Joo Ha, "Possibilities And Limitations of Applying Software Reliability Growth Models To Safetycritical Software," Journal of Nuclear Engineering and Technology, Vol. 39, No. 2, pp. 145-148, Apr 2007.

[15]. Chin-Yu Huang, Jung-Hua Lo, Sy-Yen Kuo and Michael R. Lyu, "Software Reliability Modeling and Cost Estimation Incorporating Testing-Effort and Efficiency," In Proc. of the 10th International Symposium on Software Reliability Engineering, Boca Raton, FL, pp. 62-72, Nov 1999.

[16]. Swapna S. Gokhale, Michael R. Lyu, and Kishor S. Trivedi, "Incorporating Fault Debugging Activities Into Software ReliabilityModels: A Simulation Approach," IEEE Transactions on Reliability, Vol. 55, No. 2, pp. 281-292, Jun 2006.

[17]. Katerina Goseva-Popstojanova, and Kishor S. Trivedi, "Failure Correlation in Software Reliability Models," IEEE Transactions on Reliability, Vol. 49, No. 1, pp. 37-48, Mar 2000.

[18]. Swapna S. Gokhale, Michael R. Lyu and Kishor S. Trivedi, "Software Reliability Analysis Incorporating Fault Detection and Debugging Activities," In Proc. of the Ninth International Symposium on Software Reliability Engineering, Paderborn, Germany, pp. 202-211, Nov 1998.