IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 6, Dec-Jan, 2014
ISSN: 2320 - 8791
www.ijreat.org

Capturing The Combined Effect Of Testing Time And Testing Coverage Using Two Dimensional Software Reliability Growth Models

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

***Abstract*— Software Reliability is the likelihood of breakdown 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 essential 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. 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.**

***Keywords* — Two-dimensional model, Cobb-douglas production function, Testing time and Testing Coverage.**

I. INTRODUCTION

Testing coverage plays one of the most promising roles while predicting the software reliability. Testing Coverage can be defined as a structural testing technique in which the software performance can be judged with respect to specification of the different source codes and the extent or the degree to which software is executed by the test cases. TC can help software developers to calculate the quality of the tested software and to determine the amount of additional effort needed to improve the reliability of the software besides providing customers with a quantitative confidence criterion while planning for using a software product. Hence, safety of the critical system has a high coverage objective. The basic testing coverage measures which includes different criteria such as Statement Coverage, Decision / Condition Coverage, Path Coverage and Function Coverage. Statement Coverage is defined as the proportion of lines executed in the single program. If we suppose that the different faults are uniformly distributed throughout the code, then percentage of executable statements covered shows the percentage of faults discovered[14][15]. Decision / Condition Coverage indicates whether Boolean expressions tested in control structures evaluated to both true and false. Path Coverage shows the percentage of all possible paths existing in the code exercised by the test cases. Function Coverage indicates the different proportion of functions/ procedures influenced by the software testing.

II SOFTWARE RELIABILITY GROWTH MODELS WITH TWO TYPES OF IMPERFECT DEBUGGING

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[16][17]. 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[12][13]. 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:

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

Where

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

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

III TESTING COVERAGE BASED MODELING

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

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

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[18]. 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[11]. 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)

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

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

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

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.

IV 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 [6][7]. 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[3]. 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[4][5]. 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

$$Y = AL^{\nu}K^{1-\nu} \qquad (5)$$

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.

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[9][10]. 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

$$m'(\tau) = \frac{b}{1+\beta\exp(-b\tau)}\left(N - m(\tau)\right) \qquad (6)$$

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

$$m(\tau) = \frac{N\left(1-\exp(-b\tau)\right)}{1+\beta\exp(-b\tau)} \qquad (7)$$

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

$$m(\tau) = \frac{N\left(1-\exp\left(-bs^{\alpha}u^{1-\alpha}\right)\right)}{1+\beta\exp\left(-bs^{\alpha}u^{1-\alpha}\right)} \qquad (8)$$

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[2]. 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

$$\frac{d}{dt}m(\tau) = \frac{bp}{1+\beta\exp^{-b\tau}}\left[N + xm(\tau) - m(\tau)\right] \qquad (9)$$

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

$$N(t)=\frac{N}{1-x}\left\{1-\left(\frac{(1+\beta)}{1+\beta\exp^{-bpt}}\right)^{p(1-x)}\exp^{-bpt(1-x)}\right\} \quad (10)$$

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 [8].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:

$$R(\omega/s_\pi,u_\pi)=\exp\{-[m((s_\pi+\omega),u_\pi/k)-m(s_\pi,u_\pi/k)]\} \quad (11)$$

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[1].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.

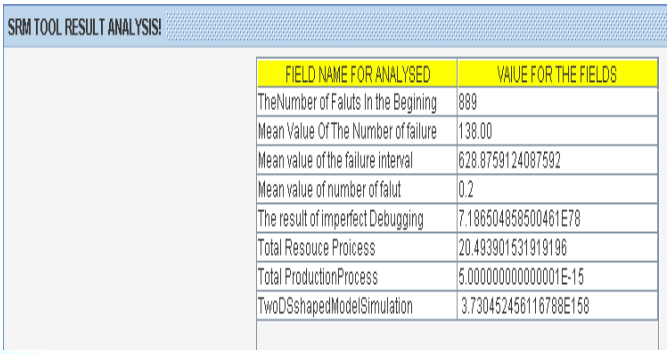

Fig. 1 Sample output of the SRGM Tool

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

$$AE=\left|\frac{M_a-a}{M_a}\right| \quad (12)$$

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

$$MSE=\frac{1}{k}\sum_{i=1}^{k}[m(t_i)-m_i]^2 \quad (13)$$

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, Yamada Rayleigh model and Huang Logistic Model are shown in the below graphs

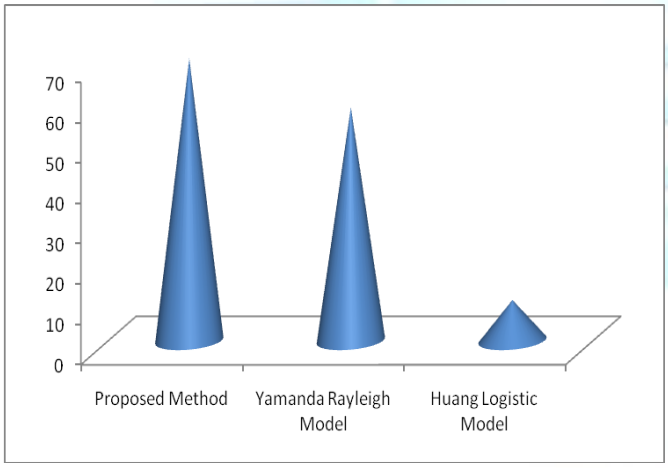

Fig.2 Comparision of AE

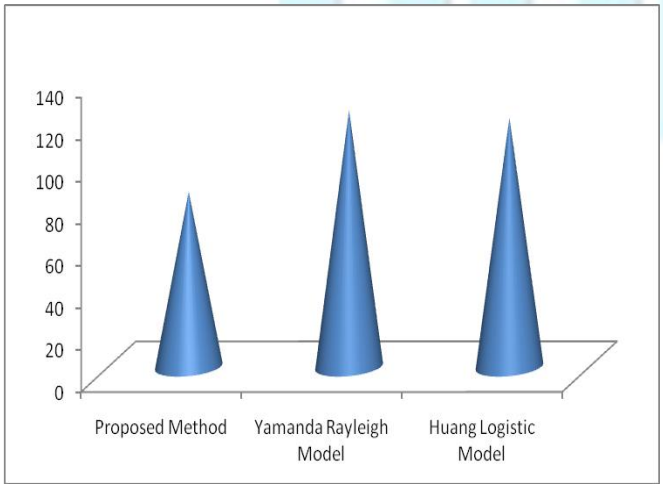

Fig.3 Comparision of MSE

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.