Software Reliability Growth Model with Varying-Time Fault Removal Efficiency As Well As With Fault Introduction

D N Gowsami[1], Anshu Chaturvedi [2], Mohammad Altaf Dar[3]

[1,3]School of Studies in Computer Science & Applications, Jiwaji University, Gwalior (M. P.), India

[2]Madhav Institute of Technology & Science, Gwalior (M. P.), India

Abstract: *A large number of software reliability growth models have been proposed to analyze the reliability of software application during the testing phase, but none of the software reliability growth model is universal to all situations.However, most of the existing software reliability growth models are developed with the assumption that all the faults detected during the testing phase are removed, and no new fault is introduced in the debugging process. As far as this assumption is considered, it seems to be unrealistic in practical; therefore it is necessary to develop a software reliability growth models with assumption that new faults are introduced when the faults in the software system are corrected and removed during testing. This study develops a new software reliability growth model incorporating both fault removal efficiency as well as fault introduction. The study considered the sense that new faults can be introduced into the software during debugging and the detected faults may not be removed completely. The fault removal is not a simple process, because detected faults would be ambiguous to find and also consumes a lot of time to remove them successfully and also a numbers of procedures are involved. The procedures involved during the removal process are fault observation, fault position and fault modification. The applicability of proposed model is shown by validating it on software failure data sets obtained from different real software development projects. The comparisons with established models in terms of goodness of fit, the Akaike Information Criterion (AIC), Sum of Squared Errors (SSE), etc. have been presented. The proposed model is compared with the growth models available in the literature, and was found encouraging.*

Keywords: Software Reliability Growth Models, faults, testing

1. Introduction

With the increasing demand of technology and complexity of the software system, the assurance of software quality becomes an essential concern. Reliability is one, and probably the most important aspect of the software quality. It usually depends on the correctness of the mapping of the system design to implementation. Thus the reliability of software is defined as the probability of failure free operation for a specified period of time in a specified environment [1]. Form early 90's, over 200 software reliability growth models have been developed to analyze the reliability of the software system but none of them is universal to all circumstances. Most of the existing software reliability growth models are developed on the assumptions that during the removal process, no new fault is introduced. From the practical point of view this assumption is not acceptable. It is known that most of the software engineers have experienced of introducing different faults in correcting a fault [2]. The study considered the sense that new faults can be introduced into the software during debugging and the detected faults may not be removed completely. There are many software reliability growth models available in the literature for example Goel-Okumoto [3], S-shaped model [4], PNZ- model [5] Ohba [6], Yamada, Ohba and Osaki [7], Zhao and Xie [8], Pham [9] and Yang [10] proposed NHPPbased Software Reliability Growth Models.

These models are developed on various assumptions, one of the common assumption is that all faults detected are corrected and removed, and no new fault is introduced. In fact, this is not true in practical point of view; however there is a chance of introducing different faults in correcting a fault.

2. Demonstration of Growth Model

2.1 Notation Used During Study

N(t)	: Total number of faults at [0,t]
m(t)	: Number of failures by time t.
a	: Initial faults content in the software
b	: Fault detection rate
R(t)	**:** Fault removal function
β(t)	**:** Fault introducing function
ω(t)	**:** Number of faults detected

2.2 Assumptions

1. Total number of faults detected N(t) follows Poisson Distribution
2. Failure rate is a function of remaining and detected faults in the software
3. New faults are introduced when the faults in the software are corrected and removed

The assumption first is widely used follows Poisson Distribution with parameters m(t), can be written as follows:

$$P(\,N(t)=n\,) = \frac{m(t)^n}{n!}\exp(-m(t)) \qquad \text{--1}$$

Where n = 0,1,2 …

By considering the above assumptions, the proposed model is described as follows

$$\frac{d\,m(t)}{dt} = b\,(\,a(t)-\omega(t)\,) \qquad \text{--2}$$

$$\frac{d\,\omega(t)}{dt} = R(t)\,\frac{d\,m(t)}{dt} \qquad \text{--3}$$

$$\frac{d\,a(t)}{dt} = \beta(t)\,\frac{d\,m(t)}{dt} \qquad \text{--4}$$

Where ω(t) is the total number of faults detected and removed successfully, m(t) represents the total number of faults, R(t) is the fault removed function and β(t) represents the fault introducing function. The initial conditions for the differential equations (2), (3) and (4) are:

$$m(0) = 0 \qquad \text{--5}$$

$$\omega(0) = 0 \qquad \text{--6}$$

$$a(0) = 0 \qquad \text{--7}$$

Hence, the total number of faults detected and removed by the time t is shown in following equation:

$$\omega(t) = a(t) - a \exp\left(b \int_0^t (\beta(\lambda) - R(\lambda))d\lambda\right) d\,t \qquad \text{--8}$$

Now, the expected number of faults can be obtained as

$$m(t) = \int_0^t exp\left(b \int_0^v (\beta(\lambda) - R(\lambda))d\lambda\right)dv \qquad \text{--9}$$

Model with Varying-Time Fault Removal As Well As Fault Introduction

So far as fault removal is considered, it is not a simple process, as it involves a series of procedures to remove them. The procedures used to remove the faults in the software are:

(i) Fault Observation
(ii) Fault Position
(iii) Fault Modification

Daniel [11] assumes the expected time to remove a fault is γ time units. Sometimes the value of γ is known due to the previous releases and other software products. Once the fault is observed, the expected number of subsequent occurrences before it is removed is γ b. Therefore fault removal efficiency can be written as:

$$\frac{1}{R} = (1 + \gamma\, b) \qquad \text{--10}$$

As far as the faults that are detected later, would be ambiguous to find and consume a lot of time to remove. However, the fault removal efficiency decreases with testing time. Fault removal efficiency is assumed to be the function of time and can be written as follows:

$$(1+Ct) \qquad \text{--11}$$

and C denotes how quickly the fault removal time can change. Then the expected number of subsequent occurrences of fault before it can be removed is

$$\gamma b(1+Ct) \qquad \text{--12}$$

Therefore, the fault removal efficiency R(t) becomes:

$$\frac{1}{R(t)} = 1 + \gamma\, b(1 + Ct) \quad \text{or}$$

$$R(t) = \frac{1}{1+\gamma\, b(1+Ct)} \qquad \text{--13}$$

Similarly fault introducing function β(t) can be written as follows:

$$\beta(t) = \beta\left(\frac{1}{1+\gamma\, b(1+Ct)}\right) \qquad \text{--14}$$

Substituting eq.13 and 14 in 8 and 9, we have:

$$m(t) = \frac{a(1+\gamma\, b)}{\beta+\gamma\, C-1}\left[\left(1 + \frac{\gamma Cbt}{1+\gamma b}\right)^{\frac{\beta-1}{\gamma\, C}-1} - 1\right] \qquad \text{--15}$$

4. Applications

In this section, applicability of the proposed model is shown by validating it on software failure data set obtained from different real software development projects. The datasets derived from different time-periods are illustrative of industrial software processes prevalent in that period. The procedure is as follows:

First, we fit the proposed model into the data i.e., parameter estimation, and obtain mean value function m(t). Secondly, the proposed model is compared with the existing models available in literature within a dataset using the SSE and AIC.

5. Parameter Estimation

To support the model applicability both the parameter estimation and model validation are the necessary aspects. The mathematical equation of the proposed SRGM is non-linear. Nevertheless, it is hard to discover the answer for a nonlinear model using Least Square Method and requires numerical algorithm to resolve it. To overcome this problem we use Statistical Software Package such as SPSS. SPSS is a statistical package for social sciences. To estimate the parameters of the proposed model, a Least Square method (Non-linear regression method) is used. Non-linear regression method finds the relationship between the dependent and independent variable. Non-linear regression can estimate models with arbitrary relationships between autonomous and dependent variable

5.1. Goodness of Fit

The term goodness of fit is used in two different contexts, in one context it denotes the question if sample of information comes from a population with a specific distribution. In another context it denotes the question of "How good does mathematical model fit to the data."

5.2. Sum of Squared Errors (SSE)

The SSE is a mathematical approach to determining the scattering of data points; found by squaring the length between each data point and the line of best fit and then summing all of the squares.The sum of the squared errors, SSE, is defined as follows:

$$SSE = \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2 \qquad \text{--16}$$

Where:
Y_i is the actual observations time series
$\hat{Y}_i$ is the estimated or forecasted time series

5.3. Akaike Information Criterion (AIC)

It is defined as AIC = -2(The value of the maximum log likelihood function) + 2(The number of the parameters used in the model).This index takes into account both the statistical goodness of fit and the number of parameters that are estimated. Lower values of AIC indicate the preferred model

5.4. Model Validation

To validate as well to determine the software reliability growth of a proposed model, it has been tried out on two testing datasets, which are documented in [12] [13] respectively. The proposed model has been compared with the NHPP of Goel Okumoto [3], S-Shaped [4]. The results are shown in tables below. After observing the goodness-of fit, the values of SSE and AIC are smaller than the values of other models. The Graph of observed values and estimated values for the datasets are illustrated in Fig. 1-2. Overall, the proposed model perform better.

Table 1: Based on the testing dataset document in [12]

Model	*SSE*	*AIC*
Goel Okumoto	$\cong 7.7*10^3$	$\cong 4.0*10^2$
S-Shaped	$\cong 5.2*10^4$	$\cong 5.5*10^2$
Proposed	$\cong 4.3*10^3$	$\cong 3.9*10^2$

Table 2: Based on the testing dataset document in [13]

Model	*SSE*	*AIC*
Goel Okumoto	$\cong 2.5*10^1$	$\cong 7.9*10^1$
S-Shaped	$\cong 2.1*10^1$	$\cong 8.5*10^1$
Proposed	$\cong 2.0*10^1$	$\cong 7.2*10^1$

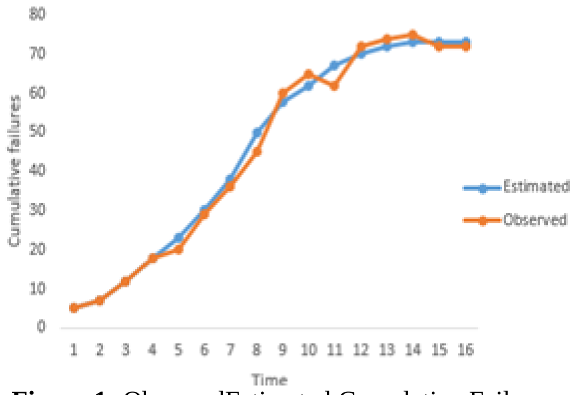

Figure 1: ObservedEstimated Cumulative Failures Curves DS-I

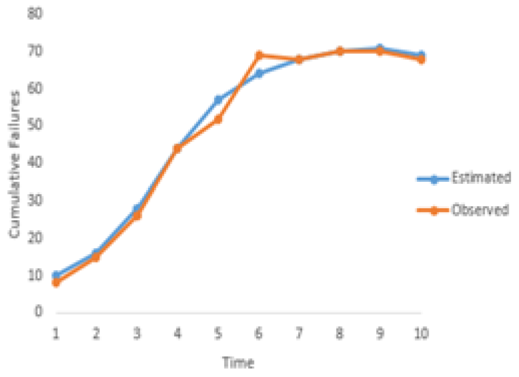

Figure 2: Observed And Estimated Cumulative Failures Curves DS-II

6. Conclusion

This paper considered the sense that new faults can be introduced into the software during debugging and the detected faults may not be removed completely. The fault removal is a complex process, because detected faults would be ambiguous to find, consumes a lot of time, and also a numbers of procedures are involved to remove to remove them. The proposed model were validated on two different datasets and are also compared with existing models in the literature base on SSE and AIC. From the comparative study it has been concluded the proposed model performs better than the other Software Reliability Growth Models.

References

[1] Musa "A Theory of Software Reliability and Its Application," IEEE Trans. on Soft. Eng., pp 312-327, 1975
[2] Yamada, Tokuno and Osaki, "imperfect debugging models with fault introduction rate for software reliability assessment," International Journal of system software, pp. 2241-2252, 1992
[3] Goel and Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," IEEE Transactions on Reliability, pp. 206–211, 1979.
[4] Yamada, Ohba, Oask, "S-shaped software reliability growth model for software detection," IEEE Transaction. On Reliability, pp 475-478, 1986.
[5] Pham, Nordmann and Zhang, "A general imperfect software debugging with S-shaped fault detection rate"IEEE Transactions on Reliability, pp. 169-175, 1999.
[6] Ohba, Osaki, and Hatoyama "Inflection S-shaped software reliability growth model, in Stochastic Models in Reliability Theory", Springer-Verlag, pp. 144-165, 1984.
[7] Yamada, Ohba, and Osaki, "S-shaped reliability growth modeling for software error detection," IEEE Transactions on Reliability, pp. 475-478, 1983.
[8] Zhao and Xie, "On maximum likelihood estimation for a general non-homogeneous Poisson process," Scandinavian Journal of Statistics, pp. 597--607, 1996.
[9] Pham and Zhang, "An NHPP software reliability models and its comparison," International Journal of Reliability Quality Safety Engineering, pp. 269-282, 1997.
[10] Yang, Sang and Lei, "An Improved NHPP Model with Time-Varying Fault Removal Delay," Journal of Electronic Science and Technology of China, pp. 334-337, 2008
[11] Daniel, Zhang and Pham, "Accounting for realities when estimating the field failure rate of software," in Pro. Of the 12th International Symposium on software Reliability Engineering, pp. 332-339, 2001.
[12] Zhang, Teng, and Pham, " Considering fault removal efficiency in software reliability assessment," IEEE Transaction on System Man and Cybernetics Part A: System and Humans, pp. 114-119, 2003
[13] Ehrlich, Prasanna, Stampfel and Wu, "Determining the cost of a stop-testing decision," IEEE Software, pp. 33-42, 1993