Advances in Computational Research
ISSN: 0975-3273 & E-ISSN: 0975-9085, Volume 4, Issue 1, 2012, pp.-74-77.
Available online at http://www.bioinfo.in/contents.php?id=33

FAULT REMOVAL EFFICIENCY IN SOFTWARE RELIABILITY GROWTH MODEL

PURNAIAH B.[1]*, RAMA KRISHNA V.[2] AND BALA VENKATA KISHORE G.[3]

Dept. of Computer Science, K.L. University, Guntur, AP, India.
*Corresponding Author: Email- [1]purnaiah@gmail.com, [2]vramakrishna2006@gmail.com, [3]g.kishore841@gmail.com.

Received: February 21, 2012; Accepted: March 06, 2012

Abstract- Software Reliability is defined as the probability of free-failure operation for a specified period of time in a specified environment. Software Reliability Growth models (SRGM) have been developed to estimate software reliability measures such as number of remaining faults, software failure rate and Software Reliability. Imperfect debugging models are considered in these models. However, most SRGM assume that faults will eventually be removed. Fault removal efficiency in the existing models is limited. This paper aims to incorporate the fault removal efficiency in software reliability growth modeling. In this paper imperfect debugging is considered in the sense that new faults can be introduced into the software during debugging and the detected faults may not be removed completely.
Key words- Non-Homogeneous Poisson process (NHPP), Software Reliability Growth Model (SRGM), Fault Removal, Maximum-Likelihood Estimation, Software Testing, Software Reliability, Software Debugging

Citation: Purnaiah B., Rama Krishna V. and Bala Venkata Kishore G. (2012) Fault Removal Efficiency in Software Reliability Growth Model. Advances in Computational Research, ISSN: 0975-3273 & E-ISSN: 0975-9085, Volume 4, Issue 1, pp.-74-77.

Introduction

over the last two decades modern society has become more increasingly dependent on hardware and software systems. Software Reliability is defined as the probability of free failure operation for a specified period of time in a specified environment. Since 1970's many SRGM have been proposed for estimation of Reliability growth of products during software development processes. SRGM are applicable to the late stages of testing in software development and can provide information in useful in predicting and improving reliability of software products. In this paper, we propose a methodology to integrate a methodology in software reliability growth model.

We are presents the formulation of the NHPP model addressing fault removal efficiency and fault introduction rate. The explicit solution of the mean value function for the proposed NHPP model is derived. This model considers the learning phenomenon Using an S -shaped fault detection rate function and introduces a constant fault introduction rate.

Software Testing is the process of exercising a program with the specific intent of finding faults prior to delivery to the users. After testing debugging is performed by programmers to discover high defects.

Software Reliability Modeling

In the family of Software Reliability models, NHPP Software Reliability models have been widely used in analyzing and estimating the reliability related metrics of software products in many applications, such as telecommunications [6],[20] etc. This model considers the debugging process as a counting process, which follows a Poisson process with a time dependent intensity function. Existing NHPP Software Reliability models can be unified into a general NHPP function proposed by pham etc.[9].The primary task of using the NHPP models to estimate Software Reliability metrics is to determine the Poisson mean, which is known as the MVF.

In this section, an NHPP model with fault removal efficiency is presented. The following are the assumptions for this model:

1. The Occurrence of software failures follows an NHPP.
2. The software failure rate at any time is a function of Fault

detection rate and the number of remaining faultsPresented at that time

3. When a software failure occurs, a debugging effort will be initiated immediately with probability ρ .The debugging is S-independent at each location of the software failures.
4. For each debugging effort, whether the fault is successfully removed, or not, some new faults may be introduced into the software system with probability $\beta(t)$

Assumption 1 is widely accepted assumption. Assumption 2 can be interpreted as follows: Software failure rate is the number of residual faults and the average failure rate of a fault. In practice, once a software failure is reported the review board members will assign a developer to look into the code. Although the fault that causes the failure may not be removed immediately, the debugging effort is still initiated. When the developer tries to modify the code new faults could be introduced to the software.

NHPP Software Reliability model with Fault Removal Efficiency

In this section, fault removal efficiency and fault introduction rate are integrated into the MVF of an NHPP model. Fault removal efficiency is defined as the percentage of bugs eliminated by reviews, inspections, and tests [5]. The MVF that incorporates both fault removal efficiency and fault introduction phenomenon can be obtained by solving the system of differential equations as follows:

$$\frac{\mathrm{dm(t)}}{\mathrm{dt}} = \mathrm{b(t)[a(t) - pm(t)} \tag{1}$$

$$\frac{\mathrm{da(t)}}{\mathrm{dt}} = \beta(\mathrm{t})\frac{\mathrm{dm(t)}}{\mathrm{dt}} \tag{2}$$

Where p represents the fault removal efficiency, which means p% of detected faults can be eliminated completed during the development process. Therefore (1), m(t) represents the expected number of faults detected by time t and pm(t) then represents the expected number of faults that can be successfully removed. Existing models usually assume that p is 100%

The marginal conditions for the differential equations (1) and (2) are as follows.

$$m(0) = 0 \tag{3}$$

$$a(0) = a \tag{4}$$

Where ***a*** is the number of initial faults in the software system before testing starts. Most existing NHPP models assume that the fault failure rate is proportional to the total number of residual faults. Equation (1) can be deduced directly from assumption 2 and 3. Software system failure rate is a function of the number of residual faults at any time and the fault detection rate (which can also be interpreted as the average failure rate of a fault). The expected number of residual faults is given by

$$x(t) = a(t) - pm(t) \tag{5}$$

Notice, that when P=1 ,the proposed model can be reduced to an existing NHPP model [17].Equation 2 can also be deduced from assumption 3 and 4.The fault current rate $a'(t)$ in software time at t is proportional to the debugging efforts to the system, which equals to $m'(t)$ because of assumption 3.Equation (5) can be used to derive explicit solutions of (1) and (2).By taking derivates on both sides of (5),we obtain

$$\frac{\mathrm{dx(t)}}{\mathrm{dt}} = \frac{\mathrm{da(t)}}{\mathrm{dt}} - \mathrm{p}\frac{\mathrm{dm(t)}}{\mathrm{dt}} = (\beta(\mathrm{t}) - \mathrm{p})\frac{\mathrm{dm(t)}}{\mathrm{dt}}$$

or

$$\frac{\mathrm{dx(t)}}{\mathrm{dt}} = (\beta(\mathrm{t}) - \mathrm{p})\mathrm{b(t)x(t)} \tag{6}$$

With marginal condition. Hence, the expected number of residual faults is given by (6) is

$$X(t) = ae^{-\int_0^t (p-\beta(\tau))b(\tau)d\tau} \tag{7}$$

From (1),the failure rate function can be expressed as follows:

$$\lambda(\mathrm{t}) = m'(t) = b(t\big(a(t) - pm(t)\big) = b(t)x(t) \tag{8}$$

Therefore, the explicit expression of the MVF can be obtained as follows:

$$m(t) = \int_0^t x(u)b(u)du = a\int_0^t b(u)e^{-\int_0^u (p-\beta(\tau)b(\tau)d\tau} \tag{9}$$

Using the result in (8),one can also obtain the solution for the fault content rate function by taking the integral of (2).The fault content rate function is given by

$$a(t) = a\left(1 + \int_0^t \beta(u)b(u)e^{-\int_0^u (p-\beta(\tau))b(\tau)d\tau}\, du\right)$$

The Reliability function based on the NHPP is, therefore

$$R\left(x/t\right) = e^{-[m(t+x)-m(t)]} \tag{10}$$

Where m(t) is given by(9).

Thus, the reliability metrics, i.e. the expected number of residual faults , software failure rate, and Software Reliability can be estimated from(7),(8) and (10) respectively.

NHPP Model

In this section, we derive a new NHPP model from the general class of model presented in the previous section. The fault detection rate function in this model, b(t) is a decreasing function with inflexion S-shaped curve[11],[12],which captures the learning process of the software developers. In the existing model [11],[12] however the upper bound of the fault detection rate is assumed to be the same as the learning curve increasing rate. This is for the purpose of calculation convenience. In this paper, we relax this assumption and use a different parameter for the upper bound of fault detection rate. The model also addresses imperfect debugging by assuming faults can be introduced during debugging with a constant fault introduction probability , β .That is

$$b(t) = \frac{c}{1+\propto e^{-bt}}$$

$$\beta(t) = \beta \tag{11}$$

Substituting (11) into (9), we obtain the MVF for the proposed model as follows:

$$m(t) = \frac{a}{p-\beta}[1 - (\frac{(1+\alpha)e^{-bt}}{1+\alpha e^{-bt}})^{(c/b)(p-\beta)}] \quad (12)$$

Note ,that the testing time t goes to infinity, m(t) converges to upper bound $\frac{a}{p-\beta}$. .The expected number of residual faults X(t) is given by

$$X(t) = a[(\frac{(1+\alpha)e^{-bt}}{1+\alpha e^{-bt}})^{(c/b)(p-\beta)} \quad (13)$$

and the software failure rate is

$$\lambda(t) = \frac{c}{1+\alpha e^{-bt}}[(\frac{(1+\alpha)e^{-bt}}{1+\alpha e^{-bt}})^{(c/b)(p-\beta)}] \quad (14)$$

Parameter Estimation and Model Comparison

Parameter Estimation: Once the analytical expression for the MVFm(t) is derived, the parameters in the MVF need to be estimated, which is usually carried out by using the maximum likelihood estimate method.

Model Comparison: Two criteria are used for model comparison. In this section, we evaluate the performance of the models using the sum of squared errors (SSE) and

$$SSE = \sum_{K=1}^{n}[Y_K - \hat{m}(t_k)]^2$$

Akaike's information criterion [1] .Both the descriptive and predictive power of the models are considered. The sum of squared error is usually used as criterion for comparison goodness of fit and predictive power.SSE can be calculated as follows:
Where

y_k Observed number of faults

$\hat{m}(t_k)$ Expected number of faults by time t_k estimated by a Model

K fault index

Another criterion used for model comparison is AIC, which can be calculated as follows.

AIC=-2*log (likelihood function at its maximum value) +2*N

Where N represents the number of parameters in the model. The AIC measures the ability of a model to maximize the likelihood function that is directly related to the degrees of freedom during fitting, increasing the number of parameters will usually result in a better fit. AIC criterion takes the degree of freedom into consideration by assigning a model with more parameters a larger penalty. The lower the SSE and AIC values, the better the model performs.

Table 1-Summary of the Software Reliability Functions and the Mean Value Functions

Model Name	Model Type	MVF (m(t))	Comments
Goel-Okumoto (G-O) [3]	Concave	$m(t) = a(1-e^{-bt})$ $a(t) = a$ $b(t) = b$	Also called exponential model.
Delayed S-shaped [21]	S-shaped	$m(t) = a(1-(1+bt)e^{-bt})$	Modification of G-O model to make it S-shaped
Inflection S-shaped SRGM [11,12]	S-shaped	$m(t) = \frac{a(1-e^{-bt})}{1+\beta e^{-bt}}$ $a(t) = a$ $b(t) = \frac{b}{1+\beta e^{-bt}}$	Solves a technical condition with the G-O model. Becomes the same as G-O if β =0
Y. Exponential [20]	S-shaped	$m(t) = a(1-e^{-r\alpha(1-e^{(-\beta t)})})$ $a(t) = a$ $b(t) = r\alpha\beta e^{-\beta t}$	Attempt to account for testing-effort
Y. Rayleigh [20]	S-shaped	$m(t) = a(1-e^{-r\alpha(1-e^{(-\beta t^2/2)})})$ $a(t) = a$ $b(t) = r\alpha\beta t e^{-\beta t^2/2}$	Attempt to account for testing-effort
Y. Imperfect debugging model (1) [22]	Concave	$m(t) = \frac{ab}{\alpha+b}(e^{\alpha t} - e^{-bt})$ $a(t) = ae^{\alpha t}$ $b(t) = b$	Assume exponential fault content function and constant fault detection rate
Y. Imperfect debugging model (2) [22]	Concave	$m(t) = a[1-e^{-bt}][1-\frac{\alpha}{b}]+\alpha a t$ $a(t) = a(1+\alpha t)$ $b(t) = b$	Assume constant introduction rate α and the fault detection rate
P-N-Z Model [16]	S-shaped and concave	$m(t) = \frac{a[1-e^{-bt}][1-\frac{\alpha}{b}]+\alpha a t}{1+\beta e^{-bt}}$ $a(t) = a(1+\alpha t)$ $b(t) = \frac{b}{1+\beta e^{-bt}}$	Assume introduction rate is a linear function of testing time, and the fault detection rate function is non-decreasing with an inflexion S-shaped model.
P-Z Model [17]	S-shaped and concave	$m(t) = \frac{1}{(1+\beta e^{-bt})}[(c+a)(1-e^{-bt}) - \frac{a}{b-\alpha}(e^{-\alpha t} - e^{-bt})]$ $a(t) = c + a(1-e^{-\alpha t})$ $b(t) = \frac{b}{1+\beta e^{-bt}}$	Assume introduction rate is exponential function of the testing time, and the fault detection rate is non-decreasing with an inflexion S-shaped model.
Proposed model	S-shaped	$m(t) = \frac{a}{p-\beta}\left[1-\left(\frac{(1+\alpha)e^{-bt}}{1+\alpha e^{-bt}}\right)^{\frac{c}{b}(p-\beta)}\right]$ $a'(t) = \beta(t)m'(t)$ $b(t) = \frac{c}{1+\alpha e^{-bt}}$ $\beta(t) = \beta$	Assume constant fault introduction rate, and the fault detection rate function is non-decreasing with an inflexion S-shaped model.

Model Evaluation and Comparison

In this section, we examine the goodness-of-fit and predictive power of the proposed model and compare it with the existing models. The first set of data is documented in Lyu [9]. We need to separate the data sets into two subsets for the goodness-of-fit test and predictive power evaluation. As seen from Table I, the proposed model provides the best fit and prediction for this data set (both the SSE and the AIC values are the lowest among all models). Furthermore, some instrumental information can be obtained from the parameter estimation provided by the proposed model.

Software failure rate can be predicted after the parameters are estimated [5]. Fig. 1 shows the trend of failure rate for the test. and post-test period. Fig. 2 illustrates the difference between the post-test failure rates predicted by several existing models listed in Table and the proposed model. For instance, the failure rate given by the G-O model is on the optimistic side, due to the following two reasons: 1) the G-O model underestimates the expected number of total faults unlike the proposed model, the G-O model does not consider the fault removal efficiency. Thus we can see that the new model has promising technical merit in the sense that it provides the development teams with both traditional reliability measures and in-process metrics.

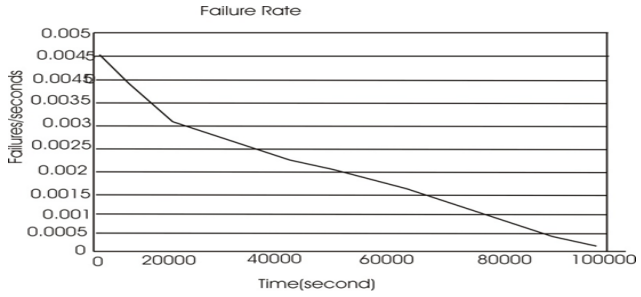

Fig.1- Failure rate for real time control data

we test the predictive power of the new model and other existing models using four sets of software failure data. Wood [13] studied eight existing NHPP models based on four data sets that stem from four major releases of software products at Tandem Computers, and found that the G-O model performs the best.. In [13], Wood used a subset of each group of the actual data to fit the models and then predicted the number of future failures. He then compared the predicted number of failures with the actual data. From the SSE values, we can see this proposed model provides a significant prediction than G-O model. The AIC value for the proposed model is also lower than that of G-O.

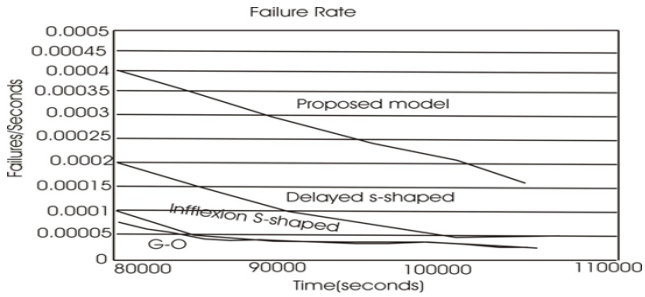

Fig. 2- Comparison of the post test failure rates by different models

Conclusion

This paper incorporates fault removal efficiency into software reliability growth model. Imperfect debugging is considered in the sense that not all fault can be removed completely, and new faults can be introduced while removing existing ones. Both the fault removal efficiency and the fault introduction function can take a time-varying form. Data collected from real applications show that the proposed model provides both, the traditional reliability measures, and also, some important in-process metrics including the fault removal efficiency and fault introduction rate. When considering reliability growth, however, the rate of evolution of the failure intensity function depends on many factors

References

[1] Akaike H. (1974) *IEEE Trans. Automat. Cont*., AC-19, 716-723.
[2] Ehrlich W., Prasanna B., Stampfel J. and Wu J. (1993) *IEEE Softw.*, 33-42.
[3] Goel A.L. and Okumoto K. (1979) *IEEE Trans. Rel.*, R-28, 206 -211.
[4] A Markovian model for reliability and other performance measures of software systems (1979) *AFIPS Conf*., 770-774.
[5] Hossain S.A. and Dahiya R.C. (1993) *IEEE Trans. Rel.*, 42, 604-612.
[6] Jeske D.R., Zhang X. and Pham L. (2001) 12*th Int. Symposium Software Reliability Engineering*.
[7] Pham H. and Zhang X. (1997) *Int. J. Rel., Quality Safety Eng*., 14(3), 269-282.
[8] Pham L. and Pham H. (2000) *IEEE Trans. Syst., Man Cybern. A*, 30, 25-35.
[9] Pham H. (2000) *Software Reliability, Springer-Verlag*.
[10] Yamada S., Ohba M. and Osaki S. (1983) *IEEE Trans. Rel.,* TR-12, 475-484.
[11] Yamada S. and Osaki S. (1985) *IEEE Trans. Software Eng*., SE-11, 1431-1437.
[12] Yamada S., Tokuno K. and Osaki S. (1992) *Int. J. Syst. Sci*., 23(12).
[13] Wood A. (1996) *IEEE Computer*, 11, 69-77.
[14] Zhang X., Jeske D.R. and Pham H., (2002) *Appl. Stochast. Models Business Ind*., 18, 87-89.
[15] Jones C.(1996) *IEEE Computer*, 29, 73-74.
[16] Kremer W. (1983) *IEEE Trans. Rel*., R-32(1), 37-47.
[17] Lyu M., Ed. (1996) *Handbook Software Reliability Engineering*.
[18] Ohba M. (1984) *IBM J. Res. Develop.*, 28, 428-443.
[19] Osaki S. and Hatoyama Y., Eds. (1984) *Reliability Theory*, *Springer-Verlag*, 144-162.
[20] Ohba M. and Yamada S. (1984) 4*th Int. Conf. Reliability Maintainability*, 430-436.
[21] Ohtera H. and Yamada S. (1990) *IEEE Trans. Rel*., 171-176.
[22] Pham H. (1993) *Idaho National Eng. Lab.*, 737.