International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075,Volume-8, Issue-7C2, May 2019

A SRGM using Time Dependent Delay Effect in Fault Detection and Removal Phenomenon

P. Agarwal, S. Rani, S. Bansal

Abstract— Many software reliability growth models are based on this approach that software faults are independent and removed immediately as they detected. But in realistic situation, some factors affect the debugging process during testing such that complicity of faults, Imperfect debugging, debugger's efficiency etc. In this paper, to fill the all these gaps a delay function is used to improve the efficiency of SRGM. A comprehensive analysis of optimal release policies based on cost and reliability is additionally provided, which could facilitate software engineers to see the most effective best optimal time to release the software package into marketplace for operational use. In the last, experimental results show the better performance of the proposed model than the other existing model.

Keywords : Non-homogeneous poisson process (NHPP), software reliability growth model (SRGM), fault detection and removal delay, fault detection and removal efficiency.

I. INTRODUCTION

With the rapid dependency on softwares, the assurance of quality and reliability has become a major task for software developers. Large software systems require regular improvement so those customers meet the updated versions by correcting the proclaimed faults. In the past three decades several software reliability growth models (SRGMs) have been submitted and many new and updated models are about to highlight in this direction. Initially, in **1979**, **Goel and Okumoto** proposed a reliability growth model which was based on perfect debugging environment. After that many more growth models were developed such as **S-shaped model (Obha, 1979), delayed S-shaped model (Yamada, 1984)** and so on which showed the very close simulation to practical software reliability engineering. It was assumed that software fault instantly removed as they were detected or isolated. In 1996**, Pham** modified the concept of debugging. He analyzed that faults are faults are removed according their severity. He categorized the faults into three categories (minor, major and critical). He gave the concept of imperfect debugging. After that many researchers improved their theory using this concept.

Recently, a couple of researchers have emphasised on the importance of lag and delay perform downside in SRGMs. **Huang and Lin (2006) and Shu et al. (2009)** studied many growth models using various debugging time lags functions for enhancing the reliability of the software. **Yang et al. (2008)** gave the theory of improved SRGM using time varying fault removal delay. **Lo and Huang (2006), Xie et al. (2007)** projected such models by that software testers can find the fault and rectify them. **Li and Pham (2017)** recommended that operating environment of software may differ with the testing environment using imperfect debugging. Recently, **Zhu and Pham (2018)** did experiment on two types of faults (dependent and Independent) and emanated that some faults are not removable for both type in removal process.

In spite of software testing, there is also one major issue in front of software managers that how much time software must be tested so that it can be released in the market at appropriate time for the operational use. If the testing process will be too long, the cost of software will also increase. So the optimal release time of the software is that when client acquires the updated version at high level of reliability but the price should be minimum. **Okumoto and Goel (1980)** considered that error detection rate is not constant and they proposed the optimal release time for this concept**.** In **1997**, **Dohi et al.** assumed that all detected errors cannot be removed immediately. These may take some extra time or efforts because of their severity then he deliberated the release policies using the delay function. **Peng et al. (2014)** suggested the optimal release policies for imperfect debugging process considering both detection and removal phenomenon. Recently a new technique of optimization for software reliability growth model was studied by **Wang et al. (2016)**.

Several SRGMs discuss that a fault is removed immediately as they are detected but in real, situations are very far from this. Removal of any fault is totally depend on many circumstances such as complexity of faults, testing methodologies, testing efforts, environment etc.. So there may be some delay to remove the fault because of these factors. This paper is organized in following sections. A new approach is carried out to explore the new dimensions of SRGMs considering fault detection and removal procedure along with delay function. Best release time based on cost (expenditure) and reliability constraint is discussed in section 3. To attest the analytical results, numerical interpretations are given in section 4. At last, outcomes of our proposed problem and future challenges are discussed in section 5.

II. SOFTWARE RELIABILITY MODELING

In this section, the model notation, assumption and formulation of the proposed model has been discussed.

Revised Manuscript Received on May 29, 2019.

P. Agarwal, SRM Institute of Science and Technology, NCR Campus, Ghaziabad, U.P. India
S. Rani, SRM Institute of Science and Technology, NCR Campus, Ghaziabad, U.P. India.
S. Bansal, SRM Institute of Science and Technology, NCR Campus, Ghaziabad, U.P. India

2.1 Nomenclature

These are some notations which will be used to define the rest of the paper.

$m_D(t)$	: Expected number of software faults detected by time t.
$m_R(t)$	: Expected number of software faults removed by time t.
$\lambda_D(t)$	: Fault detection intensity function.
$\lambda_R(t)$	: Fault removal intensity function.
a	: Total number of faults in the software.
b	: Fault detection rate.
$\beta(t)$	: Fault detection and removal efficiency.
EC(T)	: Expected cost of the software development.
C_1	: Total expected cost per unit time of testing.
C_2	: Cost of removing a fault before release (during testing).
C_3	: Cost of removing a fault before release (during operational phase, $C_2 > C_1 > 0$).
υ_0	: scale coefficient.
υ_1	: intercept value.
υ_2	: degree of opportunity loss in time.
T	: testing time of the software.
T^*	: optimal release time of the software.

2.2 Assumptions

Our model is based on the following assumptions:

- Faults detection and removal process follows the Non-homogeneous Poisson Process (NHPP).

$$\mathbf{P\{N(t) = n\} = \frac{e^{-m(t)}.m(t)^n}{n!}, n = 0, 1, 2, 3, \ldots\ldots\ldots.\infty \ldots(1)}$$

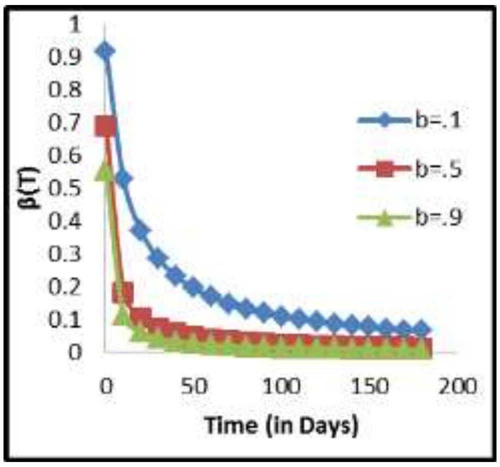
- Debugging is perfect, i.e., during detection and removal process no new faults are introduced.
- Software failure rate is a function of faults detection rate and number of remaining faults at any time t.

2.3 Model Description

Fault detection and removal is a very tedious job for the software industries. It was assumed that when a fault occurs, it is instantly removed but in reality it is not so far. Software faults first detected and then removed. Between these two processes there is time gap. This time gap is called delay effect to remove the fault. In **2001, Jeske et al.** suggested that η is the expected time to remove the detected fault and it is estimated through the experience with previous version of software testing.

The fault removal efficiency of a detected fault is

$$\mathbf{\beta(t) = \frac{1}{[1+\eta b(1+ct)]} \qquad \ldots(2)}$$



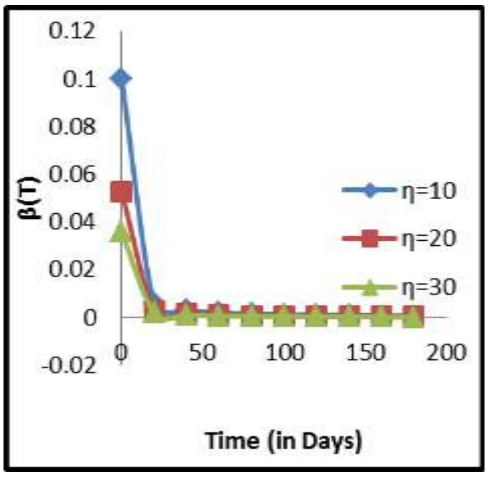


Fig. 1: Fault removal efficiency by varying (i) b (ii) η

It is shown by fig. 1 that as testing time increases, fault removal efficiency decreases. So fault removal function is supposed as delay function because when fault is detected it takes time to locate and identify then for removing. Here c measures that how fast the fault removal time is changing. According to the assumption the expected number of faults detected by time can be given by

$$\mathbf{\lambda_D(t) = \frac{d}{dt}m_D(t) = \beta(t)[a - m_D(t)] \ldots(3)}$$

$$\mathbf{\lambda_R(t) = \frac{dm_R(t)}{dt} = \beta(t)[m_D(t) - m_R(t)] \ldots(4)}$$

On solving equations (2), (3) and (4), we get

$$\mathbf{m_D(t) = a\left[1 - (1+\eta b)^k\{1 + \eta b(1 + ct)\}^{-k}\right] \ldots(5)}$$

$$\mathbf{m_R(t) = a\left[1 - (1+\eta b)^k - \frac{k(1+\eta b)^k}{[1+\eta b(1+ct)]^k}\left\{\log\frac{[1+\eta b(1+ct)]}{(1+\eta b)^{[1+\eta b(1+ct)]^k}}\right\}\right] \ldots(6)}$$

The failure intensity function for detected and removed faults is as follows:

$$\boldsymbol{\lambda_D(t) = a\left[k\eta bc(1+\eta b)^k\{1+\eta b(1+ct)\}^{-(1+k)}\right] \ldots(7)}$$

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

$$\boldsymbol{\lambda_R(t) = \frac{abck\eta(1+\eta b)^k}{\left(1+\eta b(1+ct)\right)^{1+k}}\left[k\left(1+\eta b(1+ct)\right)^k + k\left\{\log\frac{[1+\eta b(1+ct)]}{(1+\eta b)^{[1+\eta b(1+ct)]^k}}\right\} - 1\right]} \quad \textbf{(8)}$$

Where, $k = \frac{1}{\eta bc}$

III. TOTAL EXPECTED COST OF SOFTWARE

As software dependency is increasing day by day, testing process has not been only tedious job, but very expensive process also. As software becomes more advance, the rate of software raises rapidly and corresponding to it delay function majorly effects on the expenses in a system during fault detection and removal phenomenon. The postponing the release time of the software may lead to tangible and intangible losses. For resolving this issue, extra cost element is added in cost function as $C_4(T)$. The total expected cost for software development is given by

$$\mathbf{EC(T) = C_1 m_R(T) + C_2\left(m_D T_{LC} - m_R(T)\right) + C_3 T + C_4(T)} \ \textbf{...(9)}$$

Where, $C_4(T) = \nu_0(\nu_1 + T)^{\nu_2}$

IV. RELIABILITY EVALUATION

The reliability R(X/T) of the software is the feasibility that the software failure does not arise in time interval (T, T+x], i.e. software reliability does not change in the operational phase.

The reliability function for the software can be obtained as

$$\mathbf{R\left({}^{X}\!/_{T}\right) = exp[-\lambda_D(T).X]} \ \textbf{...(10)}$$

V. OPTIMIZATION PROBLEM

In this section, optimal release time of the software is discussed for the proposed growth model based on cost and reliability both constraints. Now, we formulate the optimization problem as

Minimize EC(T)
Subject to $R\left({}^{X}\!/_{T}\right) > R_0$ …(11)

If R_0 is the desired level of reliability so it can be achieved as

$$T_R = inf\left\{\lambda_D(T) \le \frac{ln\left(\frac{1}{R_1}\right)}{X} : T \in [0, T_{LC}]\right\}$$

For minimizing the cost model, differentiating equation with respect to 'T', we have

$$\frac{d}{dt}EC(T) = C_3 - (C_2 - C_1)\lambda_r(T) = 0$$

$$\therefore\ \lambda_R(T) = \frac{C_3}{C_2 - C_1}\ during\ (0, T_{LC}).$$

We can see that the second derivative of EC(T) is greater than zero, i.e., $\left[\frac{d^2EC(T)}{dT^2}\right]_{T^*=T_R}$

5.1 Optimal Release Policies Based on Cost and Reliability Constraint

P.1 : $T^* = T_R$ When $\lambda(0) > \lambda(T_R)$
P.2 : $T^* = 0$ When $\lambda(0) \le \lambda(T_R)$

VI. NUMERICAL RESULTS

In this section, a numerical illustration has been taken to calculate the expected cost EC(T) and reliability R(T) by varying the introducing faults 'a' and error detection rate 'b'. The effects of these parameters on expected cost and reliability have been examined for default parameters a=200, b=0.9, c=0.99, η=0.999, C_1=100, C_2=150, C_3=200, x=1, υ_0=.01, υ_1=0.9, υ_2=0.9. Software was tested for the 180 days and it had 200 cumulative numbers of faults. For exploring the effect of different cost elements, we consider the following sets:

Set I.	: C_1=100, C_2=150, C_3=200.
***Set II*.**	: C_1=150, C_2=150, C_3=200.
Set III.	: C_1=150, C_2=100, C_3=200.
Set IV.	: C_1=200, C_2=150, C_3=100.
Set V.	: C_1=100, C_2=200, C_3=150.

From table 1, optimal testing time for these cost data sets is analyzed and it is observed that for the 5th cost data set total expect cost is minimum comparison to other cost sets and desired reliability is achieved. So the manufacturer can release the software according their consideration.

Fig (i) and fig. (ii) illustrate the pattern of total expected cost by varying total fault content 'a' and error detection rate 'b' respectively and it is examine that cost decreases initially up to 20 days approximately and after that it increases as well as time increases.

From figs 2 (i) and 2(ii), we depict the validation of reliability with respect to time T by varying a and b respectively and it is analyzed that R(T) increases rapidly up to 80 days as time increases and after that it become almost constant (or increases very slowly).

Overall we conclude that total expected cost EC(T) initially very high due to the C_0 (testing cost is taken high) and some other parameters shows that EC(T) goes to high to keep the software maintain when extra money is spent in operational phase.

VII. CONCLUSION

In this paper, a SRGM has been proposed with the very realistic concept of fault detection and then removal phenomenon with time delay factor. It was considered that detected fault didn't remove immediately and new faults were not introduced during testing. We estimated the time dependent fault detection and removal which also increase the efficiency of the software. Experimental results also provide the optimal release time of software. The results show that the proposed model is very flexible and fits significantly better that the existing model to use and will help the professionals to improve the quality of software.

*Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication*

Time	EC(T)					R(T)
	Set 1	Set2	Set 3	Set 4	Set 5	
0	27990.71	24482.54	12813.54	20974.38	39659.72	0.123815
20	23583.94	28482.69	26220.59	32381.44	23846.04	0.198498
40	27219.16	32482.82	31585.63	35746.47	24116.35	0.621314
60	31724.06	36482.94	36080.97	38241.82	26126.03	0.803042
80	36472.76	40483.06	40332.51	40493.36	28623.3	0.883208
100	41322.33	44483.17	44483.17	42644.02	31322.33	0.923871
120	46222.88	48483.28	48582.84	44743.68	34123.33	0.946914
140	51152.62	52483.39	52653.32	46814.17	36982.7	0.961095
160	**56100.55**	**56483.5**	**56705.61**	**48866.46**	**39878.44**	**0.970385**
180	**61060.53**	**60483.61**	**60745.85**	**50906.69**	**42798.3**	**0.976776**

Table 1: Optimal testing time for different data sets for EC(T) and R(x|T*).

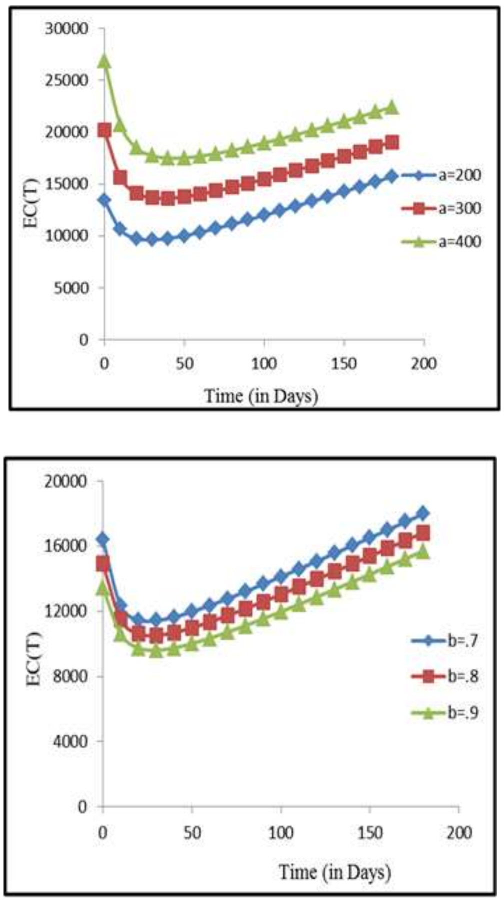


Fig. 2: Expected cost by varying (i) a (ii) b

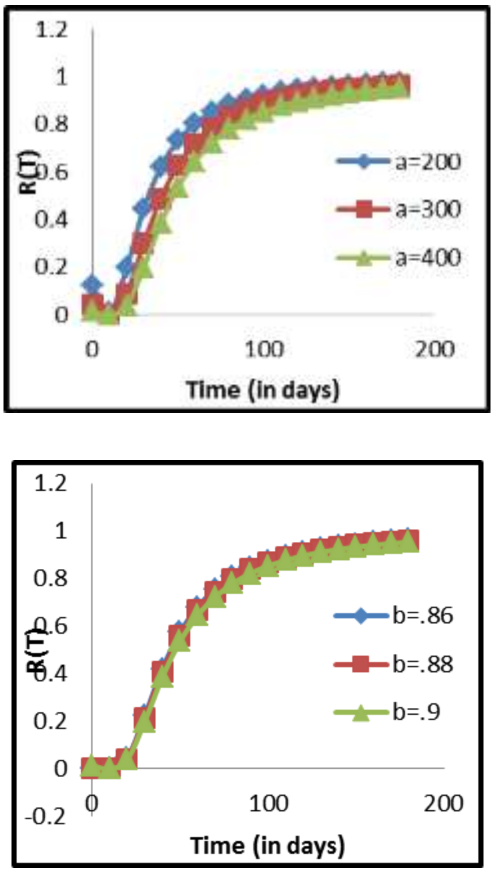


Fig. 4: Software reliability by varying (i) a (ii) b.

REFERENCES

1. **Dohi, T., Kaio, N. and Osaki, S.:** Optimal software release policies with debugging time lag, *International Journal of Reliability, Quality and Safety Engineering*, Vol. 4, No. 3, pp. 241-255, 1997.
2. **Huang, C. and Lin, C.:** Software reliability analysis by considering fault dependency and debugging time lag, *IEEE Transactions on Reliability*, Vol. 55, pp. 463-450, 2006.
3. **Li, Q. and Pham, H.:** NHPP software reliability model considering the certainty of operating environments with imperfect debugging and testing coverage, *Applied Mathematical Modelling*, Vol. 51, pp. 68-85, 2017.
4. **Lo, J. H. and Huang, C. Y.:** An integrated of fault detection and correction processes in software reliability analysis, *The Journal of Systems and Software*, Vol. 79, pp. 1312-1323, 2006.
5. **Okumoto, K. and Goel, A. L.:** Optimum release time for software systems based on reliability and cost criteria, *Journal of System and Software*, Vol. 1, pp. 315-318, 1980.
6. **Peng, R., Li, Y.F., Zhang, W.J. and Hu, Q.P.:** Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction, *Reliability Engineering & System Safety*, Vol. 126, pp. 37-43, 2014.
7. **Shu, Y., Liu, H., Wu, Z. and Yang, X.:** Modeling of software fault detection and correction processes based on the correction lag, *Information Technology Journal*, Vol. 8, No. 5, pp. 735-742, 2009.

8. **Wang, J., Wu, Z., Shu, Y. and Zhang, Z.:** An optimized method for software reliability model based on Nonhomogeneous Poisson Process, *Applied Mathematical Modelling*, Vol. 40, pp. 6324-6339, 2016.
9. **Xie, M., Hu, Q. P., Wu, Y. P. and Ng, S. H.:** A study of the modeling and analysis of software fault-detection and fault-correction processes, *Quality and Reliability Engineering*, Vol. 23, pp. 459-470, 2007.
10. **Yang, X. Sang, N. and Lei, H.:** An improved NHPP model with time varying fault removal delay, *Journal of Electronics and Technology of China*, Vol. 6, No. 3, pp. 270-273, 2008.
11. **Zhu, M. and Pham, H.:** A two phase software reliability modelling involving with software fault dependency and Imperfect fault removal, *Computer Languages, Systems & Structures*, Vol. 53, pp. 27-42, 2018.