A SOFTWARE RELIABILITY PREDICTION AND MANAGEMENT INCORPORATING CHANGE POINTS BASED ON TESTING EFFORT

ANUP KUMAR BEHERA[1] PRIYANKA AGARWAL[2]*

•

[1,2]Department of Mathematics, SRM Institute of Science and Technology, Delhi-NCR Campus, Ghaziabad, U.P. (INDIA)

[1]ab5656@srmist.edu.in, [2]priyankv@srmist.edu.in

*Corresponding Author

Abstract

This paper proposes a procedure for formulating the software reliability growth model using the non-homogeneous poisson process. We consider the software reliability growth model, which includes imperfect debugging, change points, and testing effort. Nevertheless, when formulating their software reliability models, the majority of scientists make the assumption of a constant detection rate per fault. When software is tested, they all suppose that each fault has an equal chance of being detected and that the rate is equal between generations. In practice, the fault detection rate varies depending on the test teams' abilities, program size, and software properties. Troubleshooting, even in the most realistic situations relevant to the error reintroduction rate due to incomplete debugging phenomena. In this case, changes in error detection and error introduction rates during software development program. Therefore, here we incorporate the generalized logic test workload function and change points. Parameters in software reliability modeling. Estimated using the least squares estimation method unknown parameters of the new model. Therefore, in our newly proposed model, we collect software testing data. use data from a practical application to illustrate the proposed model. Experimental results show that the proposed SRGM framework for imperfect debugging of integrated test jobs and change points has fairly accurate prediction capabilities.

Keywords: software reliability growth model, Non-homogeneous poison process(NHPP), Testing effort, Change Point

I. Introduction

Software reliability growth models (SRGMs) have a substantial historical background within the field of software engineering, serving as a pivotal tool for quantitatively evaluating and forecasting program reliability[1,2,3]. Over time, these models have developed to tackle the intricacies of software systems and the requirement for precise reliability evaluations. Many factors contribute to software failure, but mostly software fails from the design perspective. Software also fails whenever code is programmed or when changes are made to a project. Over the last few decades, numerous statistical models have been used to measure software reliability. As a result, we have discussed many established earlier models[4,5,6,7]. We believe that our new NHPP-based software reliability growth models have been proven quite efficient in practical software reliability engineering.

Researchers have examined several SRGMs throughout history to assess metrics like the number of remaining faults, software dependability, failure rate, failure intensity, and more. The literature has examined several classical models, focusing on factors such as time delay, correction procedure, fault severity, change point, and flawless debugging. Researchers have studied these models under specific assumptions. The researchers also incorporated the concept of perfect debugging, a process in which the testing team detects and fixes software errors, all while preventing the introduction of new errors during the testing process[8,9,10,11]. It is implausible that the statement is true, as the elimination process may introduce new defects that the testing team may be unaware of Several academics have suggested conducting experiments on faulty debugging.

There exist two distinct possibilities of incorrect debugging, specifically, i) imperfect fault removal and ii) generation of fault. When imperfect fault removal is present, the number of defects stays constant, signifying the elimination of the initial identified faults without introducing new ones. As new faults emerge in the system following the removal of the original problems, the overall fault content rises during the error generation scenario[12,13,14]. First introduced in 1985 was the concept of imperfect debugging. Subsequently, the concept of error production emerged, challenging the standard models' premise of complete flaw elimination upon detection. In our proposed model, we predict the existence of an imperfect debugging process that incorporates change points and testing efforts[15].

This paper fills this gap with this approach and is organized as follows. Sections II and III are discussed as NHPP software reliability growth models and software growth model change points respectively. In section IV, numerical description. Section V validates the analytical results and numerical interpretation. section VI cost model formulation and analysis of reliability and cost Section VII discusses the conclusion.

II. Non-homogeneous Poisson process uses software reliability growth models

The NHPP model is based on the assumption that software systems are subject to failures at random times due to the occurrence of residual errors. NHPP is often used to describe fault phenomena in the process testing phase. If N(t) follows a Poisson distribution with mean function m(t), then the counting process {N(t), t ≥ 0} is called NHPP with intensity function λ(t), t ≥ 0 and is given as:

$$\Pr\{N(t)=k\}=\frac{[m(t)]^k}{k!}e^{-m(t)}, k=0,1,2,... \tag{1}$$

and

$$m(t)=\int_0^t \lambda(y)dy \tag{2}$$

Inversely,

$$\lambda(t)=\frac{dm(t)}{dt} \tag{3}$$

The failure intensity function λ(t) or the mean function m(t) is the basic building block of all NHPP models.

The majority of reliability growth models for NHPP software operate under the assumption that the failure rate is directly proportional to the residual fault content. We derive a comprehensive category of NHPP-based SRGMs by solving the following differential equation:

$$\frac{dm(t)}{dt}=b(t)\left[a(t)-m(t)\right] \tag{4}$$

Where, a(t) represents the fault content function, which represents the entire number of faults in the software, including both initial and introduced faults at time t. b(t) represents the fault detection rate per fault at time t, and m(t) represents the predicted number of faults detected by time t, which is the mean value function. The solution to the differential equation (1) can be expressed as follows:

$$m(t) = e^{-B(t)}\left[m_o + \int_{t_o}^{t} a(y)b(y)e^{-B(y)}dy \right] \tag{5}$$

We have $B(t) = \int_{t_o}^{t} b(y)dy$ and $m(t_0) = m_0$, where t_0 represents the initial time of the debugging procedure. Several NHPP models that now exist can be regarded as specific instances of the overarching model described in equation (2).

I. Software reliability growth model with change point

The NHPP SRGM, which combines imperfect debugging with a change-point problem, is based on the following assumptions:

- When faults that have been detected are removed at time t, there exists the potential for the introduction of additional faults at a rate denoted as $\alpha(t)$.

$$\alpha(t) = \begin{cases} \alpha_1, \ 0 \le t \le y \\ \alpha_2, \ t > y \end{cases}$$

- We express the rate of fault detection as a step function

$$b(t) = \begin{cases} b_1, \ 0 \le t \le y \\ b_2, \ t > y \end{cases}$$

- This study proposes an NHPP model to analyze the fault detection phenomenon in software systems.

Continuous monitoring of the testing strategy and resource allocation is possible throughout the fault detection process. It may be more justifiable to reassess the provided change point (y). Based on these assumptions, we can derive the new set of differential equations to generate the new mean value function.

$$\frac{dm(t)}{dt} = b(t)\left[a(t) - m(t)\right] \tag{6}$$

$$\frac{da(t)}{dt} = \alpha(t)\frac{dm(t)}{dt}$$

$$a(0) = a, m(0) = 0$$

The mean value function of the model is given as follows:

$$m(t) = \begin{cases} \dfrac{a}{1-\alpha_1}\left[1 - e^{-(1-\alpha_1)b_1 t}\right] , & 0 \le t \le y \\ \dfrac{a}{1-\alpha_2}\left[1 - e^{-(1-\alpha_1)b_1 y - (1-\alpha_2)b_2(t-y)}\right] + \dfrac{m(y)(\alpha_1 - \alpha_2)}{1-\alpha_2}, & t > y \end{cases} \tag{7}$$

III. Proposed model

In this part, we present a software reliability growth model that integrates flawed debugging practices with change-point and testing endeavors. Commencing with the imperative nature of software reliability testing, we shall establish certain assumptions for the construction of our model. Inspection can identify a significant number of defects during the initial stage of the testing phase. Several factors, such as the efficiency of fault detection, the density of faults, the level of testing effort, and the rate of inspection, influence the pace of fault detection[17]. Subsequently, the rate of fault detection is contingent upon other supplementary characteristics, including the relationship between failures and faults, the factor of code expansion, the proficiency of test teams, the size of the program, and the testability of the software.

Assumptions of the proposed model

- According to the Non-Homogeneous Poisson Process (NHPP), the fault removal process is implemented.
- The software system may experience intermittent failures due to the presence of residual faults within the system.
- The average number of faults identified within the time interval (t, t + κt) by the present testing-effort expenditures is directly related to the average number of remaining defects in the system.
- A generalized logistic TEF model represents the testing-effort consumption curve.
$$W(t) = \frac{A}{1+n.e^{-\beta t}}$$
- When faults that have been detected are removed at time t, there exists the potential for the introduction of additional faults at a rate denoted as $\alpha(t)$.
$$\alpha(t) = \begin{cases} \alpha_1, \; 0 \le t \le y \\ \alpha_2, \; t > y \end{cases}$$
- We express the rate of fault detection as a step function
$$b(t) = \begin{cases} b_1, \; 0 \le t \le y \\ b_2, \; t > y \end{cases}$$

A generalized logistic TEF, which incorporates the fault introduction rate and change point, can characterize the software reliability growth model as follows:

$$\frac{dm(t)}{dt} * \frac{1}{w(t)} = b(t) * \left(a - m(t)\right) \tag{8}$$

$$\frac{da(t)}{dt} = \alpha(t)\frac{dm(t)}{dt}$$

$$a(0) = a, m(0) = 0$$

$W(t)$ can defined as follows:

$$W(t) = \int_0^t w(y)dy$$

The mean value function of the model is given as follows:

$$m(t) = \begin{cases} \dfrac{a}{1-\alpha_1}\left[1 - e^{-(1-\alpha_1)b_1(W(t)-W(0))}\right], & 0 \le t \le y \\ \dfrac{a}{1-\alpha_2}\left[1 - e^{-(1-\alpha_1)b_1(W(y)-W(0))-(1-\alpha_2)b_2(W(t)-W(y))}\right] + \dfrac{m(y)(\alpha_1 - \alpha_2)}{1-\alpha_2}, & t > y \end{cases} \tag{9}$$

Software reliability R(x/t) refers to the probability that no software problem will be detected during the time interval (t, t + x), where t ≥ 0, x > 0.

$$R(x f \; t) = e^{-[m(t+x)-m(t)]} \tag{10}$$

IV. Numerical Description

The existing models in the literature have employed the maximum likelihood estimation technique for parameter estimations. This study employs a nonlinear least square estimation (LSE) approach. We have utilized two historical data sets to substantiate the performance and conducted a comparative analysis between the presented model and current models. It is possible to find statistical measures like the sum of the square of error (SSE), root mean square error (RMSE), and adjusted R-square. We will now provide you with formulas for statistical measures that assess the model's fit to the data.

Root mean squared error (RMSE):

$$MSE = \frac{1}{N-n}\sum_{i=1}^{N}(y_i - m^*(t_i))^2$$

$$RMSE = \sqrt{MSE} \tag{11}$$

adjusted R^2 (Adjusted R2):

$$Adjusted\ R^2 = 1 - \frac{(1-R)(N-1)}{N-M-1} \tag{12}$$

sum of squared error (SSE):

$$SSE = \sum_{i=M}^{N}(y_i - m^*(t_i))^2 \tag{13}$$

V. Result analysis

Table 1 presents the overview of the data sets [16]. Table 2 represents the software reliability growth model and its mean value function. The unknown parameters in the proposed model are a, b$_1$, and b$_2$, and the unknown parameters in the testing effort function are β, n, and A. We have examined the proposed model with the given dataset. The results obtained from the TEF are as follows: a = 56.367, β = 12.054, and n = 0.256. The rate at which faults are introduced before the change point α_1 is evaluated to be 0.2, whereas the rate at which faults are introduced after the change point α_2 is evaluated to be 0.5. The results of the LSE reveal that the values of a, b$_1$, and b$_2$ are respectively 210.7, 0.2536, and 0.443. Table 3 shows that our proposed model is more accurate in RMSE, Adjust R^2, and SSE values than existing models. The fitting comparison of all models for using the data set is graphically illustrated in Figure 1. Figure 1, it can be seen that the proposed model fits the actual data better than all other models. Figure 2 a) RMSE, (b) Adjust R^2, (c) SSE graphically shows that compared to all models and Figure 2 shows a better fit proposed model.

Table 1: *summary of the data set*

Time	cumulative failure	testing effort consumption
1	15	2.45
2	44	4.90
3	66	6.86
4	103	7.84
5	105	9.52
6	110	12.89
7	146	17.10
8	175	20.47
9	179	21.43
10	206	23.35
11	233	26.23
12	255	27.67
13	276	30.93
14	298	34.77
15	304	38.61
16	311	40.91
17	320	42.67
18	325	44.66
19	328	47.6

Table 2: *Software reliability growth model and their mean value function*

No.	Name of the Model	MVF
1.	Goel-Okumoto (GOM)	$m(t) = a(1-e^{-bt})$
2.	Delay S-shaped (DSSM)	$m(t) = a(1-(1+bt)e^{-bt})$
3.	PNZ (PNZM)	$m(t) = \frac{a}{1+\beta e^{-bt}}\left\{[1-e^{-bt}][1-\frac{\alpha}{b}]+\alpha at\right\}$
4.	PZ (PZM)	$m(t) = \frac{1}{1+\beta e^{-bt}}\left\{[c+a][1-e^{-bt}]-\frac{a}{b-\alpha}[e^{-\alpha t}-e^{-bt}]\right\}$
5.	Proposed (PM)	Equation (9)

Table 3: *Comparison criteria*

Model	Parameter estimation	RMSE	Adjust R2	SSE
Goel-Okumoto	$a = 760.5, b = 0.03227$	12.53	0.9851	2656
Delay S-shaped	$a = 374.1, b = 0.1978$	13.73	0.9857	3205
PNZ	$a = 53.49, \alpha = 0.005551,$ $\beta = 0.38, b = 0.9413$	14.52	0.9807	3160
PZ	$a = 423.3, \alpha = 1.1, \beta = 0.8845,$ $b = 0.1167, c = 0.98$	12.24	0.9863	2246
Proposed	$a = 210.7, b_1 = 0.2536, b_2 = 0.443,$	6.1253	0.9889	1704

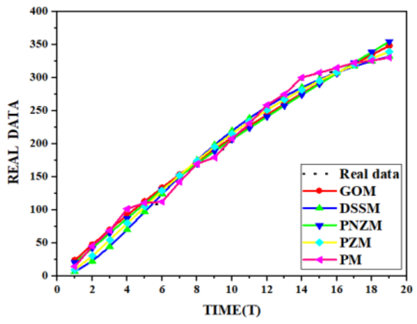
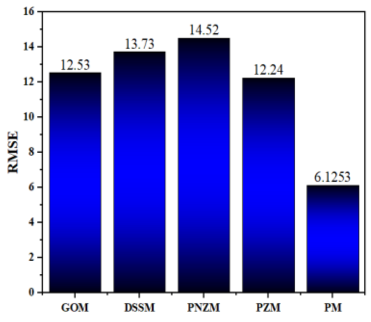
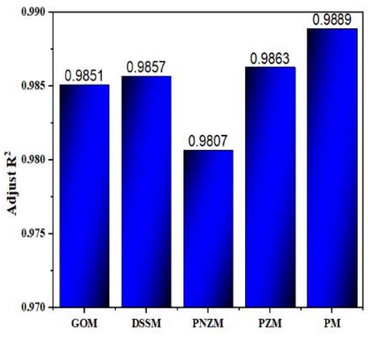
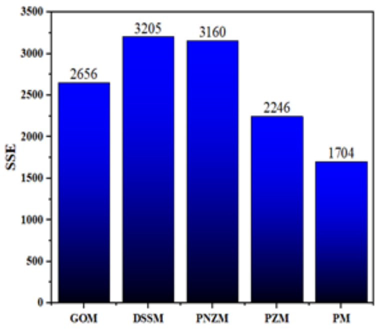


Figure 1: MVF curve for various model







(a) (b) (c)

Figure 2: *(a)RMSE, (b) Adjust R²,(c) SSE*

VI. Cost model assumptions and formulation

Based on the above assumptions, we establish a cost function.

- During the first stages of the development process [18,19], there is an incurred cost for establishing the project. Designate this as C_0.

- The cost of testing is directly proportional to the duration of the testing process. Define $E_1(T)$ as the anticipated cost of testing.

$$\text{Therefore, } E_1(T) = C_1 T^{\delta}$$

Here δ represents the discount rate.

- During debugging phase, the fault removal cost is proportional to the total time spent on debugging. Let $E_2(T)$ represent the expected cost of reducing.

$$\text{Therefore, } E_2(T) = C_2 m(T)\lambda_y$$

Here, λ_y is the expected time for resolving each defect during the testing process.

- The cost of fixing faults is directly proportionate to the time needed for their fix during the warranty period. Let $E_3(T)$ denotes the projected cost of fixing all defects during the warranty duration.

$$\text{Therefore, } E_3(T) = C_3 \left[m(T_w) - m(T) \right] \lambda_w$$

Where, λ_w represents the expected repair time for each defect during the warranty period, and T_w denotes the duration of the warranty.

- Undiscovered faults that impact the software's reliability always result in a penalty cost after its release. Let E4 (T) denote the risk cost.

$$\text{Therefore, } E_4(T) = C_4 \left[1 - R(x/T_w) \right]$$

Let E(T) be the total software expenditure. E(T) can be expressed as:

$$E(T) = C_0 + C_1 T^{\delta} + C_2 m(T)\lambda_y + C_3 \lfloor m(T_w) - m(T) \rfloor \lambda_w + C_4 \left[1 - R(x/T_w) \right] \tag{14}$$

Where, C_1, C_2, C_3 and C_4 as weights for the following: the cost of testing, the cost of error removal during testing, the cost of error removal throughout the warranty term, and the penalty for software failure.

The cost coefficients $C_0, C_1, \cdots$, etc. are often established based on prior knowledge and the current state of the market. In this study, we can take C_0 = \$100, C_1 = \$150, C_2 = \$75, C_3 = \$200, C_4 = \$1000, λ_y = 0.1, λ_w = \$0.5, x = 0.04, T_w = 40, and $\delta = 0.9$. We can determine 15 times with a development cost of \$172601.327 and reliability is 0.916. Figure 3 shows the minimum cost and this time reliability.

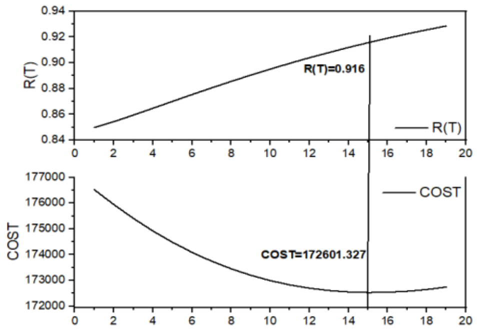


Figure 3 shows reliability and cost analysis with time

VII. Conclusion

This study introduces a novel change point software reliability growth model that incorporates the testing effort function, the NHPP framework, and imperfect debugging. In imperfect debugging, there exists a generalized logistic testing effort function and the impact of modification points.

We examine the new model and introduce the explicit mean value function. In addition, this model has been compared to several existing imprecise change point debugging models based on the root mean squared error (RMSE), adjusted R^2, and sum square error (SSE) values on the data set. Quantitative findings indicate that the suggested model has a superior level of goodness-of-fit. This proposed model appears to be slightly more advanced, but imperfect debugging, testing effort, and change point impact results in a more robust property that accurately simulates the fluctuating fault detection rate.

This study additionally examines a software cost model that integrates warranty expenses, risk costs, and mistake removal costs. This approach facilitates the determination of the optimal testing cessation point for the product, reducing anticipated total expenses, and ensuring adherence to the software reliability growth model.

By incorporating this component, the model can provide a more accurate estimation of software reliability, hence improving its practical usefulness. In our future work, we will also engage in parameter estimation using the maximum likelihood estimation technique. This strategy will offer a strong and statistically reliable method to estimate the model parameters. This property accurately represents the real-world consequences of the testing process.

References

[1] Shrivastava, Avinash K., Ruchi Sharma, and Hoang Pham. "Software reliability and cost models with warranty and life cycle." Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability 237.1 (2023): 166-179.

[2] Saraf, Iqra, and Javaid Iqbal. "Generalized multi-release modelling of software reliability growth models from the perspective of two types of imperfect debugging and change point." Quality and Reliability Engineering International 35.7 (2019): 2358-2370.

[3] Jain, M., S. C. Agrawal, and Priyanka Agarwal. "TESTING AND OPERATIONAL RELIABILITY FOR A DISTRIBUTED SOFTWARE SYSTEM WITH CORRECTION LAG CONSTRAINTS." Advances in Modeling, Optimization and Computing (2011): 630.

[4] Behera, Anup Kumar, and Priyanka Agarwal. "An SRGM using Fault Removal Efficiency and Correction Lag Function." International Journal of Reliability, Quality and Safety Engineering (2024).

[5] Rani, Sulekha, et al. "A software reliability growth model considering testing coverage subject to field environment." International Journal of Mathematics in Operational Research 18.2 (2021): 145-153.

[6] Shyur, Huan-Jyh. "A stochastic software reliability model with imperfect-debugging and change-point." Journal of Systems and Software 66.2 (2003): 135-141.

[7] Li, Qiuying, and Hoang Pham. "A testing-coverage software reliability model considering fault removal efficiency and error generation." PloS one 12.7 (2017): e0181524.

[8] Shrivastava, Avinash K., and P. K. Kapur. "Change-points-based software scheduling." Quality and Reliability Engineering International 37.8 (2021): 3282-3296.

[9] Sharma, Dinesh K., Deepak Kumar, and Shubhra Gautam. "Flexible software reliability growth models under imperfect debugging and error generation using learning function." *Journal of Management Information and Decision Sciences* 21.1 (2018): 1-12.

[10] Chatterjee, Subhashis, Deepjyoti Saha, and Akhilesh Sharma. "Multi-upgradation software reliability growth model with dependency of faults under change point and imperfect debugging." Journal of Software: Evolution and Process 33.6 (2021): e2344.

[11] Panwar, Saurabh, et al. "Software reliability prediction and release time management with coverage." International Journal of Quality & Reliability Management 39.3 (2022): 741-761.

[12] Yamada, Shigeru, Mitsuru Ohba, and Shunji Osaki. "S-shaped reliability growth modeling for software error detection." IEEE Transactions on reliability 32.5 (1983): 475-484.

[13] Tohma, Yoshihiro, et al. "The estimation of parameters of the hypergeometric distribution and its application to the software reliability growth model." IEEE Transactions on Software Engineering 17.5 (1991): 483.

[14] Goel, Amrit L., and Kazu Okumoto. "Time-dependent error-detection rate model for

software reliability and other performance measures." IEEE transactions on Reliability 28.3 (1979): 206-211.

[15] Rafi, Shaik Mohammad, and Shaheda Akthar. "Software reliability growth model with logistic-exponential testing effort function and analysis of software release policy." Proceedings of international conference on advances in computer science. 2010.

[16] Ohba, Mitsuru. "Software reliability analysis models." IBM Journal of research and Development 28.4 (1984): 428-443.

[17] Manjula, Taduru, Madhu Jain, and T. R. Gulati. "Cost optimization of a software reliability growth model with imperfect debugging and a fault reduction factor." ANZIAM Journal 55 (2013): C182-C196.

[18] Samal, Umashankar, and Ajay Kumar. "A software reliability model incorporating fault removal efficiency and it's release policy." Computational Statistics (2023): 1-19.

[19] Jain, M., P. Agarwal, and R. Solanki. "NHPP-based SRGM using time-dependent fault reduction factors (FRF) and Gompertz TEF." Decision Analytics Applications in Industry (2020): 81-89.