**Abstract**: This paper puts forth a Software reliability growth model framework that is modeled from a Non-Homogenous Poisson Process (NHPP). From the former situation to the existing situation, Software testing continues to be a paramount measure in validating the standard of a software. Test coverage measures appraise and estimate the proportion and gradation of testing in a software. Therefore, presenting a correct picture of the test-coverage becomes a prime requisite to guarantee the reliability of a software. As an enhancement over the existing models, the proposed model integrates testing coverage, error propagation and fault withdrawal efficiency, while keeping the number of parameters restrained in order to make the framework more reliable for parameter estimation. A relative analysis to assess the efficacy of the suggested model and various existing models has been carried out on the failure data obtained from four data sets.

1. **Introduction**

Software is undergoing an expeditious and rapid headway in the current times. Consequently, it becomes important for the developers to assure the quality and reliability of the software. Safety critical systems have significantly made it necessary to ensure the failure free operation of the software. Software reliability is the prospect that the system accomplishes the desired tasks under designated circumstances for a known duration of time [1]. The mathematical functions that aid to report various faults and ascertain their withdrawal are called software reliability growth models [11] . Based on this information the developers can gauge whether it is apt to make the software public or not. For the purchasers, this information approximates the confidence of using the product.

Over the time, various NHPP models have been put forth. These have been broadly bifurcated into two classes- the perfect debugging models and the imperfect debugging models [2-5]. Goel and Okumoto **[2]** ,Delayed S shaped model[3] and inflected S shaped model [4] were the earliest perfect debugging models . Later, Yamada proposed an imperfect debugging model [5]. The perfect debugging model considers that no new errors are launched during the process of debugging and the imperfect debugging model considers that during the fault withdrawal phenomenon new faults may be introduced as a part of process.

Goel and Okumoto put forward an exponential NHPP model , that considered error detection as a Non homogeneous Poisson process which has an exponentially declining rate function [2]. Later several enhancements were made and a delayed S-shaped model was proposed by Yamada and Ohba [3]. This model is considered as the learning process and it is assumed that the testing slowly refines with time. The inflected S shaped model was proposed by Ohba [4]. Ohba introduced the inflection factor and suggested that several faults cannot be identified as they remain covered by other faults.

. Yamada et. al [5] suggested an imperfect debugging model considering that new faults are often introduced when the faults originally present in the software are identified and removed during the debugging process.

The suggested model, is based upon imperfect debugging and presents an integrated approach of testing coverage, error propagation and fault withdrawal efficiency

1. **NHPP models**

A general NHPP framework is based on the below mentioned speculations [12];

1. The instances of software errors follow an NHPP
2. Debugging process commences instantly once the error is encountered.
3. Fault intensity rate is in proportion to the number of faults remaining in software at that instance.

Broadly, the software testing/debugging phenomenon is modelled as a fault counting process. A counting process {N(t), t ≥ 0} is presumed to be an NHPP with intensity function λ(t) when N(t) follows a Poisson distribution with mean function m(t) [11];

Where, m(t) is the expected number of errors detected within time (0, t) and represents the mean value function.

, Where represents fault intensity function.

1. **Model Development**

The proposed model is based on the following speculations;

1. The instances of software errors follow an NHPP and their removal also follows an NHPP
2. Debugging process commences instantly once the error is encountered and either of the two possibilities can occur;

Total number of errors is lessened by value one with probability p

Total number of errors continues to be unchanged with probability 1-p

1. Fault intensity rate is in proportion to the number of faults remaining in software at that instance.
2. During the debugging process new faults are brought in with probability constant α.

Consider that *c*(*t*) is the proportion of the code that has been inspected up to time *t*. Therefore, *c*(*t*) is a growing function of testing time *t*. Generally, it increases rapidly from the inception of testing process as many test cases are available and after certain time point, when the number of test cases starts to lessen; the testing coverage’s increasing rate lowers and eventually *c*(*t*) becomes flat . Thus, a concave / S-shaped function can be chosen to model the testing coverage function. Evidently, (1-*c*(*t*)) showcases, the proportion of code that has not been brought under inspection up to time interval *t*.  *c*′(*t*) which is the derivate of the coverage function *c*(*t*), denotes the coverage rate. Consequently, the function *c*′(*t*)/(1-*c*(*t*)) is proposed to be used to indicate the fault detection rate , which is generally taken as the standard speculation by Software reliability growth models in view of testing coverage[20].

On the basis of above supposition, the mean value function encompassing fault withdrawal efficiency and test- coverage can be obtained by solving the below equation [10,20] :

(1)

Here, *a*(*t*) indicates the comprehensive fault content function of the software, parameter  *p* indicates the fault withdrawal efficiency, i.e.  *p*% of the errors that are spotted ; can be eliminated completely during the evolving process of the software, *β* (beta) signifies the constant of proportionality, *m*(*t*) indicates the expected number of faults detected up to time interval *t*, and *pm*(*t*) indicates the expected number of errors that can be banished wholly. Consequently, [*a*(*t*)-*pm*(*t*)] constitutes the expected number of the left-over errors that are present in the software at time interval *t*.

In, case of the already existing models p is taken as 100 percent [6].

As discussed earlier ,the total fault content function *a*(*t*), is a linear function of the testing time and can be represented by [16,10];

;

where a indicates the original fault number existing in the software before the onset of testing process.

Substituting the value of a(t) into (1) , and considering the initial condition that at t = 0,  mean value function m(t) = 0, we get;

(2)

 c(0) indicates the coverage function when t = 0.

As, already discussed the coverage function is a non-negative and non-declining function of testing time t [16]. That is, the testing coverage function may exhibit S-shaped behaviour which is suitable to be reported by a concave curve of the form ;

Substituting the value of the coverage function c(t) in the mean value function ;

c(0)=1 – (1+b(0)) \* e0

c(0)= 1- (1+0)1

c(0)=1-1 = 0

substituting the value of c(0) and c(t) in above equation

Therefore, the mean value function for the propounded model is;

On that account, it must be observed that the suggested model integrates testing coverage, error propagation and fault withdrawal efficiency.

**Data analysis and model comparisons**

In this section, the efficacy of the proposed model has been assessed. In order, to estimate the parameters, Statistical package for social sciences (SPSS) has been employed. Four datasets have been used from the published papers and relative analysis of the suggested model and the existing models has been performed on these data sets.

|  |  |  |
| --- | --- | --- |
| Dataset 1 | Failure data of Application program dataset | [8] |
| Dataset 2 | Failure data of Tandem software (release 2) | [21] |
| Dataset 3 | Failure data of Monitor and control system | [7] |
| Dataset 4 | Failure data of wireless system | [23] |

Table 2

Summary of m(t) of various models.

|  |  |
| --- | --- |
| Model name | MVF |
| Goel – Okumoto model |  |
| Delayed S- shaped model |  |
| Inflected S- shaped model |  |
| Yamada imperfect debugging model | A,b,α >0 |
| Proposed model | A,b,p,,β >0 |

**Model Comparison**

The comparison criteria that have been used include MSE (mean square error) , R square [5] , Adjusted R square [3].

Table 3

Relative analysis on various failure data sets and models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DATASET | MODEL | R2 | Adjusted R2 | MSE |
| Dataset 1 | G-O model | 0.986 | 0.98425 | 156.264 |
| Dataset 1 | Delayed S-shaped model | 0.984 | 0.982 | 188.517 |
| Dataset 1 | Inflected model | 0.992 | 0.9904 | 98.211 |
| Dataset 1 | Imperfect model | 0.987 | 0.9844 | 162.896 |
| Dataset 1 | Proposed model | 0.989 | 0.9847 | 155.674 |
| Dataset 2 | G-O model | 0.982 | 0.9799 | 26.002 |
| Dataset 2 | Delayed S-shaped model | 0.990 | 0.9888 | 14.688 |
| Dataset 2 | Inflected model | 0.995 | 0.994 | 7.127 |
| Dataset 2 | Imperfect model | 0.984 | 0.981 | 25.365 |
| Dataset 2 | Proposed model | 0.991 | 0.9877 | 16.758 |
| Dataset 3 | G-O model | 0.965 | 0.9643 | 804.517 |
| Dataset 3 | Delayed S-shaped model | 0.985 | 0.9847 | 331.964 |
| Dataset 3 | Inflected model | 0.987 | 0.9866 | 300.140 |
| Dataset 3 | Imperfect model | 0.975 | 0.9742 | 578.873 |
| Dataset 3 | Proposed model | 0.985 | 0.9842 | 340.950 |
| Dataset 4 | G-O model | 0.986 | 0.9832 | 1.375 |
| Dataset 4 | Delayed S-shaped model | 0.986 | 0.9832 | 1.375 |
| Dataset 4 | Inflected model | 0.994 | 0.992 | 0.715 |
| Dataset 4 | Imperfect model | 0.990 | 0.9866 | 1.133 |
| Dataset 4 | Proposed model | 0.993 | 0.9888 | 0.984 |

The final results attained show reasonably good performance of the proposed model. It must be noted that the parameters incorporated in the suggested model are only five. In general, when more parameters are encompassed by any framework, it is more pliant and flexible for the data distribution, but on contrary it becomes less reliable for the parameter estimation [12].

**Conclusions**

In this research, we propounded a new software reliability growth model in order to present the correct picture of the testing coverage and integrated it with other prime measures like fault withdrawal efficiency and error generation. The existing models discussed in the paper, did not give the accurate picture of the testing coverage measure which is a prerequisite for a system to be reliable. We, restrained the number of parameters to five, so as to make the system more reliable for the parameter estimation. The results showcased a fairly good performance of the suggested model.

References

1. Lyu MR. Handbook of Software Reliability Engineering. New York: McGraw-Hill; 1996
2. (Goel and Okumoto, 1979)
3. Yamada S, Ohba M, Osaki S. S-Shaped Reliability Growth Modeling for Software Error Detection. IEEE Transactions on Reliability. 1983; 32(5):475–84
4. Ohba M. Inflection S-Shaped Software Reliability Growth Model. Stochastic Models in Reliability Theory. Berlin: Springer,1984.144–62
5. YAMADA S, TOKUNO K, OSAKI S. Imperfect debugging models with fault introduction rate for software reliability assessment. Int. J. Syst. Sci. 1992; 23(12):2241–52.
6. Pham H, Zhang X. An NHPP Software Reliability Model and Its Comparison. International Journal of Reliability Quality & Safety Engineering. 1997; 14(3):269–82.
7. Tohma Y, Yamano H, Ohba M, Jacoby R. The estimation of parameters of the hypergeometric distribution and its application to the software reliability growth model. IEEE Transactions on Software Engineering. 1991; 17(5):483–9
8. Ohba M. Software Reliability Analysis Models. Ibm Journal of Research & Development. 1984; 21(4): 428–43.
9. Pham, Hoang, and Xuemei Zhang. "NHPP software reliability and cost models with testing coverage." *European Journal of Operational Research* 145.2 (2003): 443-454.
10. Li, Qiuying, and Hoang Pham. "A testing-coverage software reliability model considering fault removal efficiency and error generation." *PloS one* 12.7 (2017): e0181524.
11. Kapur, P. K., et al. *Software reliability assessment with OR applications*. London: Springer, 2011.
12. Chiu, Kuei-Chen, Yeu-Shiang Huang, and Tzai-Zang Lee. "A study of software reliability growth from the perspective of learning effects." *Reliability Engineering & System Safety* 93.10 (2008): 1410-1421.
13. Ohba, Mitsuru. "Software reliability analysis models." *IBM Journal of research and Development* 28.4 (1984): 428-443.
14. 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.
15. Wood, Alan. "Predicting software reliability." *Computer* 29.11 (1996): 69-77.
16. Pham, Hoang, and Xuemei Zhang. "NHPP software reliability and cost models with testing coverage." *European Journal of Operational Research* 145.2 (2003): 443-454.
17. Kapur, P. K., Anu G. Aggarwal, and Abhishek Tandon. "Two dimensional software reliability growth model with faults of different severity." *Communications in Dependability and Quality Management* 13.4 (2010): 98-110.
18. Malaiya, Yashwant K., et al. "Software reliability growth with test coverage." *IEEE Transactions on Reliability* 51.4 (2002): 420-426.
19. Park, Joong-Yang, Gye-Min Lee, and Jae-Heung Park. "A general coverage-based NHPP SRGM framework." *Communications for Statistical Applications and Methods* 15.6 (2008): 875-881.
20. Kapur, P. K., et al. "Two dimensional multi-release software reliability modeling and optimal release planning." *IEEE Transactions on Reliability* 61.3 (2012): 758-768.
21. Wood, Alan. "Predicting software reliability." *Computer* 29.11 (1996): 69-77.
22. Data set wireless