A Software Reliability Growth Model Considering Mutual Fault Dependency

Md. Asraful Haque, Nesar Ahmad
•
Department of Computer Engineering, Z.H. College of Engineering & Technology,
Aligarh Muslim University, Aligarh-202002, India,
md_asraf@zhcet.ac.in , nesar.ahmad@gmail.com

Abstract

Many software reliability growth models (SRGMs) have been introduced since 1970s. Most of the models consider that the faults are independent and debugging method is perfect. In this paper, we present a new SRGM under the assumption that the faults are mutually dependent i.e. repairing a detected fault may introduce new faults or it may simultaneously correct some future faults without any additional effort. The model is validated on two real datasets that are widely used in many studies to demonstrate its applicability. The comparisons with eight established models in terms of Mean Square Error (MSE), Variance, Predictive Ratio Risk (PRR) and R^2 have been presented.

Keywords: Software Reliability, SRGM, Software Testing, Debugging, Fault Prediction, Project Management.

I. Introduction

Today we are very much dependent on software systems in many facets of our life. The demand of highly reliable software has rapidly increased. Software development is a time consuming and intensive job that involves many people, process and technology. Thus software systems are error prone. Reliability is an end-user quality feature related to the system-usage. Software reliability can be defined as the probability that no failure occurs up to a specified time interval. Unlike hardware, it is not possible to measure or quantify software reliability directly. With the help of probabilistic and statistical methods, different approaches have been developed for measuring software reliability. However, use of software reliability growth models (SRGMs) is a popular and traditional way to describe the failure patterns and predict the reliability. The SRGMs are represented in abstract forms that include many parameters based on certain assumptions. In the last four decades, a sufficient number of SRGMs have been suggested at regular intervals [1][2]. They are broadly divided into two groups: times between failure models and fault count models [3][4]. The models that recognize MTBF (mean time between failures) as input are referred to as times between failures models and the models that use failure rate are referred to as fault count models. Examples of some times between failure models are The Jelinski-Mornada de-eutrophication model (in short J-M model), Littlewood-Verral model etc. The J-M model, known as one of the earliest models, assumes that the failure rate is constant between failures and reduces in fixed step-size following the repair of each fault [5]. The Littlewood-Verral model which is an

updated version of the J-M model considers that the times between failures follow an exponential distribution [6]. Most of the SRGMs fall under the category of fault count models such as follows. Goel-Okumoto model (or G-O model) is a Non Homogeneous Poisson process (NHPP) with an exponentially decaying rate function [7]. Musa Okumoto model represents the cumulative number of failures over time in terms of a logarithmic function [8]. Yamada et. al. proposed the delayed S-Shaped model to describe the increase-decrease failure rate pattern considering the learning process of the testers' skills [9]. Ohba suggested the Inflection S-shaped model with the concept of mutual fault dependency (i.e. some faults are discoverable only after the detection of some specific faults) [10]. Yamada et. al. [11] also suggested a two variant Imperfect Debugging Model by modifying G-O model that incorporates the linear fault introduction rate. H.Pham et. al. suggested PNZ model by considering fault introduction rate is a linear function of testing time [12] and PZ model by considering fault introduction rate is an exponential function of testing time [13]. Recent studies in reliability modelling include different approaches of machine learning techniques or deal with the issue of uncertainties due to random operating environment. Jaiswal and Malhotra [14] tested different ML techniques for software reliability prediction on different datasets collected from industrial projects and compared the results. They concluded that adaptive neuro fuzzy inference system (ANFIS) is the most effective method compared to others in predicting software reliability. Chang et. al. [15] proposed a testing-coverage model considering the uncertainty of operating environment. Pham [16] discussed two NHPP models with and without considering the uncertainty factor based on a log-log distribution function.

Till date, near about 200 software reliability growth models have been suggested [4] and most of them are based on the assumption that the faults are independent. This assumption is not true in real testing environment. The paper presents a model that considers the issue of dependent faults.

II. Proposed Model

A generalized failure intensity function of a software reliability growth model under the assumption that the fault detection rate is proportional to the number of remaining faults is given by [17]:

$$\frac{dm(t)}{dt} = b(t)[a - m(t)] \quad (1)$$

where,

- m(t) : The mean value function (Expected number of faults detected by time t).
- a : Total expected number of faults that exist in the system.
- b(t) : Time dependent fault detection rate per fault.

In practice, it is seen that faults are dependent. Sometimes repairing one fault introduces new faults. Sometimes repairing one fault removes some future faults without any extra effort. Therefore, number of fault detections differs with the number of fault removals. Let us consider that p is the fault removal rate per detected fault. Therefore, number of faults removed at time t is $pm(t)$ and number of remaining faults will be $(a - pm(t))$. From (1), we can write,

$$\frac{dm(t)}{dt} = b(t)[a - pm(t)] \qquad \text{where, } p>0; \quad (2)$$

If p<1 then it means imperfect debugging with high fault introduction rate. If p>1then it indicates the one to many mapping between fault detection and fault removal. P =1 represents perfect debugging with one to one mapping. The solution of eqn. (2) for the mean value function m(t) with the initial condition m(0) = 0, is given by:

$$m(t) = \frac{a}{p}\left(1 - e^{-p\int_0^t b(t)dt}\right) \quad (3)$$

We also assume that the fault detection rate per fault will increase with time. Initially the testing team takes time to understand the behavior of the system; hence, fault detection rate is relatively slow. As testing progresses the team gradually becomes familiar with the system leading to higher fault detection rate. In this paper, we consider the following function of b(t):

$$b(t) = b(1+ct) \quad (4)$$

c -is a parameter that reflects the change in fault detection rate with time and b is a constant. Replacing the value of (4) in eqn. (3),

$$m(t) = \frac{a}{p}\left(1 - e^{-pb(t+ct^2/2)}\right) \quad (5)$$

This is the mean value function of the proposed model. Now we can derive the failure intensity function from (5),

$$\lambda(t) = \frac{dm(t)}{dt} = ab(ct+1)e^{-pb(t+\frac{ct^2}{2})}) \quad (6)$$

III. Analysis of the Model

We evaluate the performance of the proposed model on two different datasets (DS1 and DS2) and compare the results with the following eight existing models (Table 1).

Table 1. *Software Reliability Models*

Model	m(t)
Goel-Okumoto Model [7]	$a(1-e^{-bt})$
Delayed S-Shaped [9]	$a(1-(1+bt)e^{-bt})$
Inflection S-shaped [10]	$\frac{a\left(1-e^{-bt}\right)}{1+\beta e^{-bt}}$
Yamada Imperfect Model-1 [11]	$\frac{ab}{\alpha+b}\left(e^{\alpha t}-e^{-bt}\right)$
Yamada Imperfect Model-2 [11]	$a\left(1-e^{-bt}\right)\left(1-\frac{\alpha}{b}\right)+\alpha at$
P-N-Z Model [12]	$\frac{a\left(1-e^{-bt}\right)\left(1-\frac{\alpha}{b}\right)+\alpha at}{1+\beta e^{-bt}}$
Testing Coverage Model [15]	$N\left(1-\left(\frac{\beta}{\beta+(at)^b}\right)^{\alpha}\right)$
Loglog Fault-detection Rate Model [16]	$N(1-e^{-(a^{t^b}-1)})$
Proposed Model	$\frac{a}{p}(1-e^{-pb(t+ct^2/2)})$

A. Comparison Criteria

None of the SRGMs is reliable to get accurate results in all circumstances and thus be selected a priori. It is necessary to compare multiple models and then select the one that match the failure data most accurately. There are many standard criteria known as "Goodness of Fit" criteria available for model comparison and selection [18-20]. In this study, we have used the following four criteria.

- MSE: The mean square error (MSE) is a calculation of how far the estimated values vary from the actual observations, and is defined as [18][19]:

$$\text{MSE} = \frac{\sum_{i=1}^{n}(m_i - m(t_i))^2}{n-k}$$

- Variance: The variance is the standard deviation of the differences between actual and predicted data. It is defined as [18][19]:

$$\text{Variance} = \sqrt{\frac{\sum_{i=1}^{n}(m_i - m(t_i) - \text{Bias})^2}{n-1}}$$

$$\text{where, Bias} = \frac{\sum_{i=1}^{n}(m(t_i) - m_i)}{n}$$

- PRR: The predictive-ratio risk (PRR) measures the per estimate model deviation from the actual data and is defined as [18][19]:

$$\text{PRR} = \sum_{i=1}^{n}\left(\frac{(m(t_i) - m_i)}{m(t_i)}\right)^2$$

- R^2: It measures how well a model fits the data. It is also known as the "coefficient of determination" and defined as [18][19]:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(m_i - m(t_i))^2}{\sum_{i=1}^{n}(m_i - \sum_{j=1}^{n}\frac{m_j}{n})^2}$$

The smaller values of MSE, Variance, PRR and AIC criteria indicate fewer numbers of fitting errors and better performance [20] whereas the value of R^2 is expected to be 1 for an ideal model.

B. Dataset Description

The basic approach of the SRGMs is to predict the future faults by analyzing the past failure data. The performance of an SRGM greatly depends on the type of datasets. We consider two datasets from Tandem Technical Report-96.1 [21][22] in our experiment. The Tandem report contains four failure datasets related to the four different releases of Tandem Computer Project. Table 2 presents a failure dataset (DS1) of release 1 having 100 software faults collected over the 20 weeks of testing and the 10000 hours of execution. Table 3 provides the dataset (DS2) of Release 4 having 42 faults collected over the 19 weeks of testing and the 11305 hours CPU execution.

Table 2. *DS1: Tandem computers failure data – Release 1*

Test Week	CPU hrs	Cumulative Faults	Test Week	CPU hrs	Cumulative Faults
1	519	16	11	6539	81
2	968	24	12	7083	86
3	1430	27	13	7487	90
4	1893	33	14	7846	93
5	2490	41	15	8205	96
6	3058	49	16	8564	98
7	3625	54	17	8923	99
8	4422	58	18	9282	100
9	5218	69	19	9641	100
10	5823	75	20	10000	100

Table 3. *DS2: Tandem computers failure data – Release 4*

Test Week	CPU hrs	Cumulative Faults	Test Week	CPU hrs	Cumulative Faults
1	254	1	11	7621	32
2	788	3	12	8783	32
3	1054	8	13	9604	36
4	1393	9	14	10064	38
5	2216	11	15	10560	39
6	2880	16	16	11008	39
7	3593	19	17	11237	41
8	4281	25	18	11243	42
9	5180	27	19	11305	42
10	6003	29	-	-	-

C. Parameter Estimation

The parameters of all the 9-models mentioned in Table 1, have been estimated using the least square estimation (LSE) technique and time weeks. The resultant values of the parameters have been provided in Table 4 for the datasets DS-1 and DS-2 respectively.

Table 4. *Parameter Estimation using LSE*

Model	DS1	DS2
Goel-Okumoto	a = 130.2, b = 0.083	a = 89.63, b = 0.037
Delayed S-Shaped	a = 104, b = 0.265	a = 47.23, b = 0.207
Inflection S-shaped	a =110.829, b =0.172, β =1.205	a =43.36, b =0.279, β = 6.459
Yamada Imperfect Model-1	a =130.2, b =0.083, α =4.25*10^{-4}	a =87.94, b =0.037, α= 0.0001
Yamada Imperfect Model-2	a =130.2, b =0.083, α =1.283*10^{-4}	a =87.69, b =0.038, α=0.0001
P-N-Z Model	a =116.324, b = 0.14, α = 0.001, β = 0.787	a =31.44, b = 0.353, α= 0.023, β = 7.275
Testing Coverage Model	N = 119.205, a = 13.798*10^{-3}, b = 1.111, α = 65.069, β = 7.337	N = 44.398, a = 0.04, b =1.672, α = 25.908, β = 5.356
Loglog Fault-detection Model	N =105.109, a =1.095, b = 0.947	N = 48.72, a = 1.051, b = 1.237
Proposed Model	a = 100.926, b = 0.087, p = 0.937, c = 0.092	a = 41.598, b = 0.027, p = 0.967, c = 0.659

D. Results and Comparison

The criteria values (MSE, Variance, PRR and R^2) of all the models have been provided in Table 5 and 6. For both the datasets, the proposed model provides highest R^2, lowest Variance and PRR and second lowest MSE values. The findings clearly indicate that the proposed model fits better than many existing models studied in the paper. Figure 1 and 2 display two curves representing the deviation of the measured faults according to the proposed model from the actual observed faults for DS-1 and DS-2 respectively.

Table 5. *Model Comparison for DS1*

Model	MSE	Variance	PRR	R^2
Goel-Okumoto	12.915	3.511	0.203	0.986
Delayed S-Shaped	28.065	5.772	1.084	0.969
Inflection S-shaped	10.564	3.177	0.305	0.989
Yamada Imperfect Model-1	13.787	3.514	0.204	0.986
Yamada Imperfect Model-2	13.688	3.504	0.203	0.986
P-N-Z Model	12.662	3.315	0.277	0.988
Testing Coverage Model	14.577	3.445	0.3	0.987
Loglog Fault-detection Rate Model	8.437	2.861	0.238	0.991
Proposed Model	10.688	3.064	0.295	0.990

Table 6. *Model Comparison for DS2*

Model	MSE	Variance	PRR	R^2
Goel-Okumoto	5.1	2.527	6.726	0.976
Delayed S-Shaped	1.095	1.017	0.126	0.995
Inflection S-shaped	1.117	0.999	0.8	0.995
Yamada Imperfect Model-1	5.380	2.222	6.324	0.976
Yamada Imperfect Model-2	5.410	2.519	6.816	0.976
Testing Coverage Model	1.50	1.340	0.111	0.995
Loglog Fault-detection Rate Model	3.75	1.951	5.235	0.983
P-N-Z Model	1.086	0.973	0.424	0.995
Proposed Model	1.150	0.983	0.340	0.995

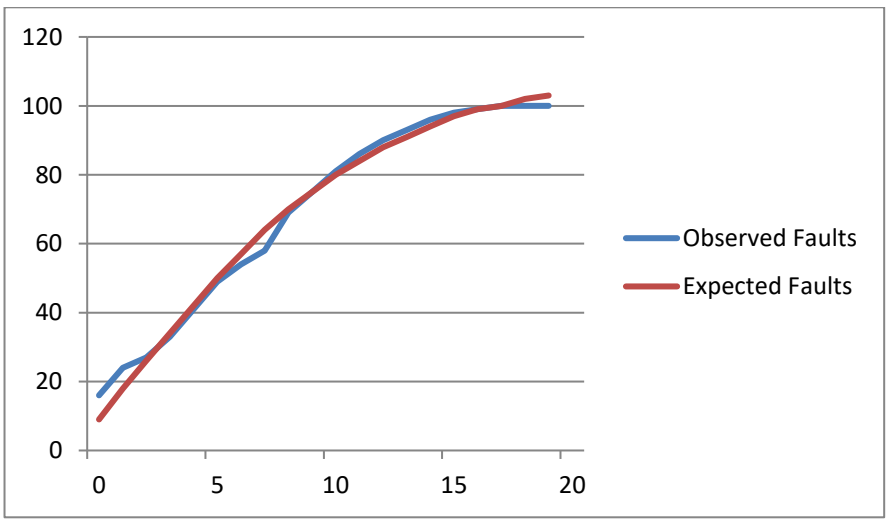

Figure 1: *Expected faults vs. observed faults for DS1*

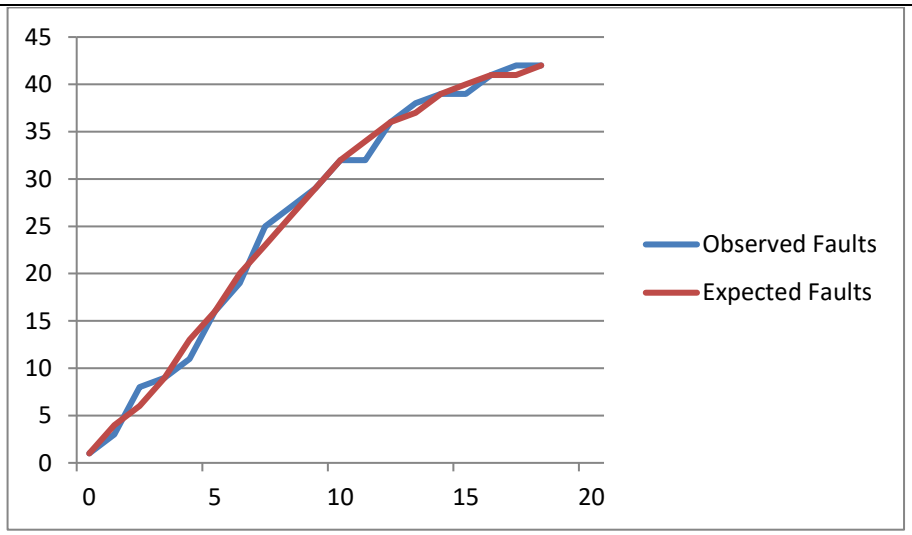

Figure 2: *Expected faults vs. observed faults for DS2*

IV. Conclusion

The paper presents a new software reliability growth model addressing the issue of mapping between fault detection and fault removal processes. The proposed model incorporates a time dependent fault detection rate function. The model has been tested with two actual failure datasets and compared with eight established models using four different criteria. The results are very promising. However, there are some scopes for possible improvements. We only tested the model with two datasets, which is insufficient to claim any superiority about the model performance. Moreover, the datasets are relatively old. Future work will focus on broader validation of the proposed model based on more recent datasets considering different comparison criteria.

Acknowledgement

This paper is a part of PhD work of Md. Asraful Haque, supported by Visvesvaraya PhD Scheme, MeitY, Govt. of India < MEITY-PHD-2980>.

References

[1] John D. Musa, “Software Reliability Engineering”, McGraw-Hill, 1999, ISBN: 9780079132710.
[2] Q. Li, H. Pham, “A testing-coverage software reliability model considering fault removal efficiency and error generation” PLoS ONE 12(7):e0181524, 2017.
[3] R. Lai, M. Garg, “A Detailed Study of NHPP Software Reliability Models”, Journal of Software, Vol. 7, No. 6, p. 1296-1306, June 2012.
[4] M. A. Haque and N. Ahmad, “Key Issues in Software Reliability Growth Models”, Recent Advances in Computer Science and Communications, 13: 1, 2020, DOI: 10.2174/2666255813999201012182821.
[5] Z. Jelinski, P. B. Moranda, “Software reliability research”, in Statistical Computer Performance Evaluation, Freiberger W. (ed.), Academic Press, New York, p. 465–484, 1972.
[6] B. Littlewood, J. L. Verall, “A Bayesian Reliability Growth Model for Computer Software”, Journal of Royal Statistical Society Series C-Applied, 22(3), pp. 332-345,1973.
[7] A. L. Goel, K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," in IEEE Transactions on Reliability, vol. R-28, no. 3, pp. 206-211, 1979.
[8] J. D. Musa, K. Okumoto, “A logarithmic poisson execution time model for software reliability measurement”, In Proceedings of the 7th international conference on Software engineering, IEEE Press, Piscataway, NJ, USA, 230-238, 1984.

[9] S. Yamada, M. Ohba and S. Osaki, "s-Shaped Software Reliability Growth Models and Their Applications," in IEEE Transactions on Reliability, vol. R-33, no. 4, pp. 289-292, Oct. 1984.
[10] M. Ohba, "Inflection S-Shaped Software Reliability Growth Model", In: Osaki S., Hatoyama Y. (eds) Stochastic Models in Reliability Theory. Lecture Notes in Economics and Mathematical Systems, vol 235. Springer, Berlin, Heidelberg. 1984.
[11] S. Yamada, K. Tokuno, and S. Osaki, "Imperfect debugging models with fault introduction rate for software reliability assessment," Int. J. Syst. Science, vol. 23, no. 12, 1992.
[12] H. Pham, L. Nordmann, and X. Zhang, "A general imperfect software debugging model with s-shaped fault detection rate," IEEE Trans. Reliability, vol. 48, pp. 169–175, 1999.
[13] H. Pham H, X. Zhang X, "An NHPP Software Reliability Model and Its Comparison", International Journal of Reliability, Quality & Safety Engineering, 14(3): 269-82, 1997.
[14] A. Jaiswal, R. Malhotra, "Software reliability prediction using machine learning techniques", Int J Syst Assur Eng Manag 9, 2018, 230–244.
[15] I. H. Chang, H. Pham, S.W. Lee, K.Y. Song, "A testing-coverage software reliability model with the uncertainty of operation environments", Int. J. Syst. Sci. Oper. Logist. 2014, 1, 220–227.
[16] H. Pham, "Loglog fault-detection rate and testing coverage software reliability models subject to random environments", Vietnam Journal of Computer Science 1(1), pp. 39–45, 2014.
[17] P. K. Kapur, H. Pham, S. Anand, K. Yadav, "A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation", in IEEE Transactions on Reliability, vol. 60, no. 1, pp. 331-340, March 2011, doi: 10.1109/TR.2010.2103590.
[18] K. Sharma, R. Garg, C. K. Nagpal, R. K. Garg, "Selection of Optimal Software Reliability Growth Models Using a Distance Based Approach," in IEEE Transactions on Reliability, vol. 59, no. 2, pp. 266-276, 2010.
[19] M. Anjum, MA Haque, N. Ahmad, "Analysis and Ranking of Software Reliability Models based on Weighted Criteria Value", Int. Journal of Information Technology and Computer Science, vol.5, no.2, page: 1-14, 2013.
[20] K.Y. Song, I.H. Chang, H. Pham, "An NHPP Software Reliability Model with S-Shaped Growth Curve Subject to Random Operating Environments and Optimal Release Time", Applied Sciences 2017, 7, 1304.
[21] A. Wood, "Software reliability growth models", In Tandem Technical Report-96.1, 1996.
[22] A. Wood, "Predicting software reliability," in Computer, vol. 29, no. 11, pp. 69-77, Nov. 1996.