STATISTICS IN TRANSITION-new series, December 2009
Vol. 10, No. 3, pp. 457—464

COMBINED EFFECT OF FAULT DETECTION AND FAULT INTRODUCTION RATE ON SOFTWARE RELIABILITY MODELLING

S. Chatterjee[1], L.N. Upadhyaya[2], J.B. Singh [3]and S. Nigam [4]

ABSTRACT

This paper proposes a software reliability growth model to study the combined effect of increasing error detection and decreasing error introduction rate under imperfect debugging. The model is developed based on non homogeneous Poisson process (NHPP) and can be used to estimate and predict the reliability as well as the cost of a software product. Some real life data has been used to validate the proposed model and to show its usefulness. Comparison of this model with other has been carried out.

Key words: Software Reliability, Imperfect Debugging, Multiple Failures, Non Homogeneous Poisson Process, Increasing Fault Detection Rate, Decreasing Fault Removal Rate.

Nomenclature

N (t) – counting process representing the cumulative number of errors
Poim (m(t)) – Poisson process with mean m(t)
A – mean initial error content in the software
b_i – fault detection rate per type i fault, i=1,2,3,…,k
p_i – content proportion of type i fault, i=1,2,3…,k
$\lambda_i(t)$ – fault detection rate per unit time of type i error, i=1,2,3,…,k
$\lambda(t)$ – fault detection rate per unit time
n(t) – number of errors to be eventually detected
β – error introduction rate
$m_i(t)$ – expected number of type i errors by time t, i=1,2,3,….,k
T – total test time i.e., release time
s_j – cumulative time

[1] Assistant Professor, Dept. of Applied Mathematics, ISMU, Dhanbad-826004, Corresponding author. Email: chatterjee_subhashis@rediffmail.com.
[2] Professor, Dept. of Applied Mathematics, ISMU, Dhanbad-826004.
[3] Research Scholar, Dept. of Applied Mathematics, ISMU, Dhanbad.
[4] Project Fellow, Department of Applied Mathematics, ISMU, Dhanbad.

R(x/t) – conditional reliability of software
mle – maximum likelihood estimate

Introduction

Computers are widely used in various fields of life including business and safety critical systems. Application of computer means the application of software. Therefore, there exists an increasing demand of highly reliable software. Since assessment of software reliability is one of the major concern in present-day software industries, one need a good mathematical model to estimate the reliability of software. Research in the area of software reliability has been going on since last three decades. Detail studies related to software reliability has been presented in [1,2,3]. Initially, various software reliability models were developed using the concept of perfect debugging process. Some of the important software reliability models among them are [4,5,6,7,8,9,10,11,12,13]. Latter stage various software reliability models were developed using the concept of imperfect debugging. Some of them also throws light to other aspects of software reliability studies like: estimation of cost, release time etc.[14,15,16,17,18,19,20,21,22].

Present-day software development process has became very complex. Therefore, there is a need to develop an efficient mathematical model which can give better prediction of software reliability and take care of different aspects of software development process as well. Among these aspects of software development process, error detection rate, i.e., FDR and error introduction rate, i.e., FIR during software debugging plays very crucial role in the growth of software reliability.

Pham[19] developed a software reliability model considering imperfect debugging and presence of multiple failures. He considered FDR and FIR both as constant and different for different types of errors. Also, he considered presence of three types of errors: type 1 errors, i.e., critical errors – which are very difficult to detect, type 2 errors, i.e., major errors – which are difficult to detect, type 3 errors, i.e., minor error – which are easy to detect. Software testing as well as debugging are done by human being. As time progress test personnel learns more about the software. As a result, FDR increases and FIR decreases with respect to time. In this paper the FIR β and FDR b are being considered as a function of cumulative time and a more realistic software reliability model has been developed to study the combined effect of increasing FDR and decreasing FIR on the growth of software reliability. As error introduction and detection depends on the knowledge of the test personnel, these factors cannot be different for different types of errors. Due to this reason, FDR has been considered the same for all types of errors and the same is the case for FIR. Also, in this paper, presence of k types of errors (for generalization) is considered. The proposed model is based on non homogeneous Poisson process and some real life data are used to validate the model.

Model development

A non homogeneous Poisson process based model considering imperfect debugging and presence of k types of error has been proposed in this section.

Model Assumptions

The following assumptions are made for the development of the model.

(i) During removal of detected errors, it is possible to introduce new errors.

(ii) The probability of finding an error in software is proportional to the number of remaining errors in the software.

(iii) There exist k types of errors in software.

(iv) The error detection process follows a non homogeneous Poisson process.

(v) Fault detection rate (FDR) b is same for k types of errors and increases with respect to cumulative time. Here b is considered as a logistic function, i.e., $b = \dfrac{1}{1+(\dfrac{1}{\alpha_0}-1)e^{-rs_i}}$, where, α_0 and r are constant, s_j is the cumulative time.

(vi) Fault introduction rate (FIR) β is same for k types of errors and decreases with respect to cumulative time, where $\beta = \dfrac{1}{1+s_j^2}$

For mathematical simplification the cumulative time s_j is considered here. Otherwise, analysis will be more complicated. According to the assumption (iv), the model has been formulated as $\Pr\{N(t)=n\} = poim(n, m(t)), n = 0,1,2.....$ To obtain the reliability of software the mean value function $m(t)$, i.e., the expected number of software failures to be determined. It is obtained by solving the following differential equations.

$$\frac{dm_i(t)}{dt} = \lambda_i(t) \quad , \quad \frac{dm_i(t)}{dt} = b[n_i(t) - m_i(t)]$$

$$\frac{dn_i(t)}{dt} = \beta \ \frac{dm_i(t)}{dt}, \ m(t) = \sum_{i=1}^{k} m_i(t) \ \text{ where } n_i(0) = ap_i, \ m_i(0) = 0$$

Solutions of the above differential equations give the expression for $m(t), \lambda(t)$ and $R(x/t)$ as follows.

$$m(t) = \sum_{i=1}^{k} \frac{ap_i[1-e^{-(1-\beta)bt}]}{(1-\beta)} \quad , \quad \lambda(t) = \sum_{i=1}^{k} ap_i be^{-(1-\beta)bt} \text{ and}$$

$$R(x/t) = e^{-[m(t+x)-m(t)]} = e^{-\sum_{i=1}^{k} m_i(x)e^{-(1-\beta)bt}}$$

Results & Discussion

To demonstrate the usefulness of the proposed model and for comparison with Pham model [19], presence of three types of errors, i.e., k=3 in a software is considered. Three types of errors are: type 1 errors, i.e., critical errors – which are very difficult to detect, type 2 errors, i.e., major errors – which are difficult to detect, type 3 errors, i.e., minor error – which are easy to detect. For illustration purpose the failure data of Misra [23] given in Table-I are used. Here $p_1 = 0.0173,\ p_2 = 0.342,\ p_3 = 0.6407$. The values of the constant r, α_0 are 1.5 and 0.05 respectively.

Table-I. Original failure data

Test Week	Test Hours	Critical errors	Major errors	Minor errors	Test Week	Test Hours	Critical errors	Major errors	Minor errors
1	62.5	0	6	9	20	25	0	2	3
2	44	0	2	4	21	12	0	1	1
3	40	0	1	7	22	55	0	3	2
4	68	1	1	6	23	49	0	2	4
5	62	0	3	5	24	64	0	4	5
6	66	0	1	3	25	26	0	1	0
7	73	0	2	2	26	66	0	2	2
8	73.5	0	3	5	27	49	0	2	0
9	92	0	2	4	28	52	0	2	2
10	71.4	0	0	2	29	70	0	1	3
11	64.5	0	3	4	30	84.5	1	2	6
12	64.7	0	1	7	31	83	1	2	3
13	36	0	3	0	32	60	0	0	1
14	54	0	0	5	33	72.5	0	2	1
15	39.5	0	2	3	34	90	0	2	4
16	68	0	5	3	35	58	0	3	3
17	61	0	5	3	36	60	0	1	2
18	62.6	0	2	4	37	168	1	2	11
19	98.7	0	2	10	38	111.5	0	1	9

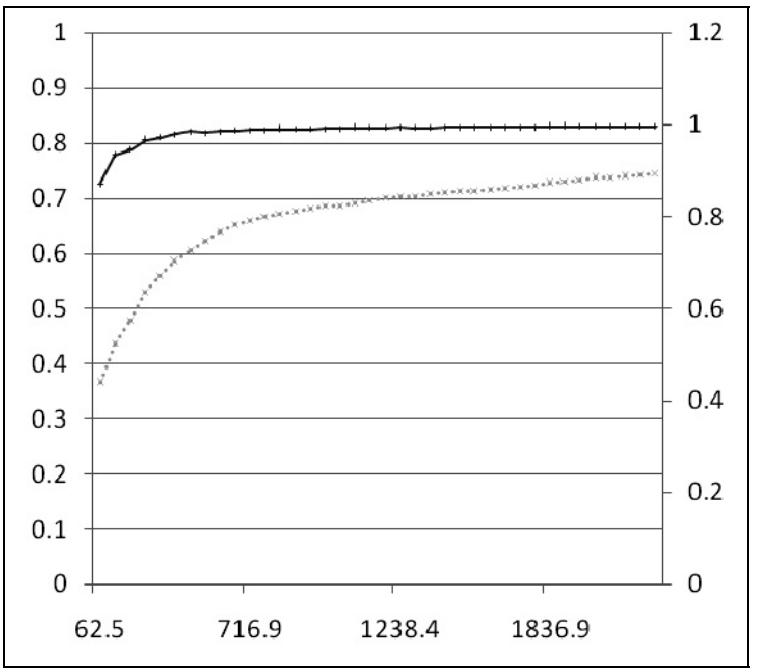
To estimate the error a, mle is used. Solving the mle equation, the estimated value of errors obtained is, $\hat{a} = 42$. While Pham [19] estimated the total number of errors using his model as $\hat{a} = 0.428$. Though the estimated total number of

errors $\hat{a} = 42$ using the proposed model is less than the actual errors present in the software, still it is better than the result obtained in [19]. Better results can be obtained using appropriate function for b and β. Conditional reliability $R(x/T)$ has been computed using the equation $R(x/t) = e^{-[m(t+x)-m(t)]}$ considering x=0.2. $R(x/T)$ and FDR b at each cumulative time are given in Table 2. The corresponding graph is given in Fig. 1.

Table 2. FDR & Conditional reliability corresponds to cumulative time

Failure Time	FDR (b)	R(x/T)	Failure Time	FDR (b)	R(x/T)
62.5	0.438	0.871	1226.4	0.842	0.9931
106.5	0.524	0.9345	1238.4	0.844	0.9938
146.4	0.575	0.9462	1293.4	0.845	0.9934
214.5	0.635	0.966	1342.4	0.849	0.9936
276.4	0.672	0.9727	1406.4	0.852	0.9939
342.4	0.702	0.9795	1432.4	0.855	0.994
415.5	0.727	0.9862	1496.4	0.857	0.9943
489	0.748	0.9828	1547.4	0.860	0.9945
581	0.769	0.9856	1599.4	0.863	0.9946
652.4	0.782	0.9876	1669.4	0.865	0.9949
716.9	0.792	0.9883	1753.9	0.869	0.9951
781.6	0.801	0.9892	1836.9	0.872	0.9953
817.6	0.806	0.9897	1896.9	0.876	0.9955
871.6	0.812	0.9903	1969.4	0.880	0.9956
911.1	0.817	0.9907	2059.4	0.884	0.9958
979.1	0.823	0.9914	2117.4	0.885	0.9959
1040.1	0.824	0.9918	2177.4	0.887	0.9961
1102.7	0.829	0.9923	2345.4	0.892	0.9963
1201.4	0.835	0.9929	2456.9	0.895	0.9965

Figure 1. Fault Detection Rate and Conditional Reliability w.r.t. Cumulative time



Conclusion

In this paper an attempt has been made to study the combined effect of increasing FDR and decreasing FIR on error estimation as well as reliability growth of software when the debugging process is imperfect. It has been observed that the assumptions made about the FDR and FIR are logical. The model can be used in other software failure data by assuming proper FDR & FIR. The proposed model can be used for estimating release time and cost of software.

Acknowledgement

Authors acknowledges University Grants Communication (UGC), New Delhi, India, for financial help in the project number F.No.33-115/2007(SR). Also, authors acknowledge ISM, Dhanbad, for providing facilities to carry out the work.

REFERENCES

[1]. MUSA, J.D.; IANNINO, A. and OKUMOTO, K.; "Software Reliability Measurement, Prediction, Application", *McGraw-Hill Int. Ed*., 1987

[2]. XIE, M.; "Software Reliability Modelling", *World Scientific Press*, 1991.

[3]. LYU; M.R.; "Handbook of Software Reliability Engineering", McGraw-Hill: NY, 1996.

[4] JELINSKI, Z. and MORANDA, P.B.; "Software Reliability Research", Statistical Computer Performance Evaluation**"**, *W. Freiberger. Ed. Academic, N.Y*., 1972, p. 465–484.

[5] SHOOMAN, M.L.; "Probabilistic Models for Software Reliability Prediction", *Statistical Computer Performance Evaluation, W. Freiberger, Ed., Academic, N.Y*., 1972, p. 485–502.

[6] WAGNOR, W.L.; "The Final Report on Software Reliability Measurement Study", *Report TOR-0074 (4112)-1, The Aerospace Corporation, El Segundo, C.A*., 1973.

[7] SCHICK, G.J. and WOLVERTON, R.W.; "An Analysis of Competing Software Reliability Model*", IEEE Trans. On Software Eng.*, vol. SE-4, 1978, p. 104–120.

[8] MUSA, J.D.; "A Theory of Software Reliability and Its Application", *IEEE Trans. On Software Eng*., vol. SE-1, 1975, p. 312–327.

[9] LITTLEWOOD, B. and VERRALL, J.L.; "A Bayesian Reliability Growth Model for Computer Software", Appl. *Statist*., vol. 22, 1973, p. 332–346.

[10] SINGPURWALLA, N.D. and SOYER. R.; "Assessing (Software) Reliability Growth Using A Random Co-efficient Autoregressive Process and Its Ramifications", *IEEE Trans. On Software Eng*., vol. SE-11, 1985, p. 1456–1464.

[11] XIE, M.; "A Shock Model for Software Reliability", Microelectron. *Rel*., vol. 27, 1987, p. 717–724.

[12] GOEL, A.L. and OKUMOTO, K.; "A Time-Dependent Error Detection Rate Model for Software Reliability and Other Performance Measure", IEEE *Trans. On Re*l., vol. R-28, 1979, p.206–211.

[13] CHATTERJEE, S; MISRA, R.B. and ALAM, S.,S.; "Joint Effect of Test Effort and Learning Factor on Software Reliability and Optimal Release Policy"; *International Journal of System Science*; Vol. 28; No. 4; 1997; p. 391–396.

[14] SUMITA, U. and SHANTIKUMAR J.G.; "A Software Reliability Model With Multiple-Error Introduction & Removal", *IEEE Trans. On Rel*., vol. R-35, 1986, p. 459–462.

[15] FAKHRE - ZAKERI, I. and SLUD, E.; "Mixture Models for Reliability of Software With Imperfect Debugging: Identifiably of Parameters" *IEEE Trans. On Rel*., vol. 44, 1995, p. 104–113.

[16] ZEEPHONGSEKUL, P.; XIA, G. and KUMAR, S.; "Software-Reliability Growth Model: Primary Failures Generate Secondary-Faults Under Imperfect Debugging", *IEEE Trans. On Rel*., vol. 43}, 1994, p. 408–413.

[17] KAREER, N.; KAPUR, P.K. and GROVER, P.S.; "A S-Shaped Software Reliability Growth Model With Two Types of Error", *Microelectron. Re*l., vol. 3, 1985, p.1085–1090.

[18] XIA, G.; ZEEPHONGSEKUL. P. and KUMAR, S.; "Optimal Software Release Policy With a Learning Factor for Imperfect Debugging", *Microelectron. Rel*., vol. 33, 1993, p. 81–86.

[19] PHAM, H.; "A Software Cost Model With Imperfect Debugging Random Life Cycle and Penalty Cost", Int*. J. of Sys. Sci*., vol. 27, 1996, p. 455–463.

[20] KAPUR, P.K.; SHARMA, K.D. and GARG. R.B.; "Transient Solutions of a Software Reliability Model With Imperfect Debugging and Error Generation", *Microelectron. Rel*., vol. 32, 1992, pp. 475–478.

[21] CHATTERJEE, S; MISRA, R.B. and ALAM, S.,S.; "A Generalized Shock Model for Software Reliability"; C*omputer and Electrical Engineering-An International Journal,* Vol. 24; 1998, p.no.: 363–368.

[22] ZHANG, X; TENG, X; and PHAM, H.; "Considering Fault Removal Efficiency in Software Reliability Assessment", *IEEE Trans. On Systems Man and Cybernetics-Part A: Systems and Humans*, Vol. 33, No. 1, Jan., 2003, p. 114–120.

[23] MISRA, P.N.; "Software Reliability Analysis", *IBM Syst. J*., Vol. 22, 1983, p. 262–272.