Cost optimization of a software reliability growth model with imperfect debugging and a fault reduction factor

Madhu Jain[1] T. Manjula[2] T. R. Gulati[3]

(Received 19 December 2013; revised 12 May 2014)

Abstract

In modern society people depend on both hardware and software systems. A software system is embedded in every activity of a computer system. The desired performance of a software system is an important issue for many critical systems. Over the past decades, many software models were proposed for estimating the growth of reliability. To improve software quality, software reliability growth models (SRGM) play an important role. The present investigation deals with a SRGM with imperfect debugging, change points and a fault reduction factor (FRF). A FRF is the net number of faults removed in proportion to the failures experienced. This article proposes a new scheme for constructing a SRGM based on a non-homogeneous Poisson process by

http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/7834
 Published June 21, 2014, as part of the Proceedings of the 11th Biennial Engineering Mathematics and Applications Conference. ISSN 1446-8735. (Print two pages per sheet of paper.)

considering a constant FRF. The main focus is to provide an efficient parametric decomposition for a SRGM. Numerical examples are given to illustrate the validity of analytical results.

Contents

1 Introduction C183

2 Model description C185

3 Estimation of parameters C187

4 Software release time based on reliability criteria C188
4.1 Software release time based on cost requirement C189
4.2 Software release time based on reliability and cost requirement C189

5 Performance measures for goodness of fit C190
5.1 Mean square error C190
5.2 Accuracy of estimation C190

6 Numerical results C191

7 Conclusion C193

References C193

1 Introduction

The goal of every software industry is to develop software which is error and fault free. To improve the software quality, software reliability engineering plays an important role in many aspects throughout the software life cycle. Software reliability assessments such as the number of errors, failure rate,

reliability requirements and total system developing costs are appropriate criteria for deciding when to stop software testing and when to release it. Therefore, the modelling of software reliability and accurately predicting trends is essential for determining overall reliability of software. Numerous software reliability growth models (SRGMs) were developed during the last four decades [11, 17, 7, 13] and still new models are being explored. One of the most important parameters which controls the growth of software reliability is the fault reduction factor (FRF) proposed by Musa [9]. A FRF is defined as the net number of faults removed in proportion to the failures experienced [10]. Hsu et al. [4] studied the time variable FRF for enhancing software reliability but without a change point. We study an imperfect debugging SRGM with constant FRF and a change point. The optimal software release time problem based on minimizing cost subject to achieving a given level of reliability is studied. The comparison criteria of goodness of fit is also performed.

Whenever software is developed it is possibile to introduce new errors during the development/testing phase. Sometimes faults cannot be removed perfectly because of the complexity of the faults. This phenomenon is called imperfect debugging. Previous studies [13, 16, 18, 2] incorporated the concept of imperfect debugging. Recently, Ahmad et al. [1] analysed a SRGM by considering the log-logistic testing effort and imperfect debugging.

In practice the failure distribution is affected by many factors, such as the testing strategy, running environment and resource allocation. Changing these factors during the software testing phase is called a change point. Shyur [16] studied a SRGM with imperfect debugging and a change point. The environmental effects of a change point on the SRGM were evaluated by Zhao et al. [20]. Kapur et al. [6] examined a SRGM for analyzing errors of different severity using a change point. Li et al. [19] developed a sensitivity analysis of release times of SRGM, incorporating the testing effort with multiple change points.

The major issue for project managers is to decide when to release the software. The optimal software release time is the time that customers get the software

at minimum cost and a high level of reliability. Okumoto and Goel [12] proposed the optimum release time for software systems based on reliability and cost criteria. The optimal testing resource allocation during module testing considering cost, testing effort and reliability was evaluated by Jha et al. [5]. Later, Quadri and Ahmad [14] developed an optimal release policy of a SRGM by considering the Weibull testing effort function. Quadri et al. [15] described SRGM with generalized exponential testing effort and optimal software release policies. Our main focus is to calculate the optimal release time of the software by using cost-reliability criteria. The effects of parameters which have the most significant influence on the cost function are examined.

2 Model description

Most SRGM focus on the software testing phase where software defects are detected, isolated and removed. The proposed model is motivated by the work of Hsu et al. [4]. In this section, a SRGM based on a non-homogeneous Poisson process (NHPP) is developed by incorporating imperfect debugging, change point and FRF concepts. In the model, constant FRF is used to characterize the effect of environmental factors on the testing process. During the software testing, the fault detection rate $b(t)$ and fault introduction rate $\beta(t)$ may change at some instant in time $t = \tau$.

The fault detection rate function with change point is defined as

$$b(t) = \begin{cases} b_1, & 0 \leqslant t \leqslant \tau, \\ b_2, & t > \tau. \end{cases} \tag{1}$$

The fault introduction rate during testing with change point is

$$\beta(t) = \begin{cases} \beta_1, & \leqslant t \leqslant \tau, \\ \beta_2, & t > \tau. \end{cases} \tag{2}$$

The following assumptions are made for modelling purposes.

1. The error elimination process follows the NHPP.
2. The software system is subject to failure at random times caused by remaining faults in the system.
3. All faults in a program are mutually independent.
4. The mean number of faults detected in the interval $(t, t + \Delta t]$ is proportional to the mean number of remaining faults in the system.
5. When detected errors are removed, it is possible to introduce new errors.
6. The fault detection rate is proportional to a constant FRF.

Based on the assumptions, we have the following differential equations for the mean value function $m(t)$ of fault detection:

$$\frac{dm(t)}{dt} = b(t)B[a(t) - m(t)], \tag{3}$$

$$\frac{da(t)}{dt} = \beta(t)\frac{dm(t)}{dt}, \tag{4}$$

where $a(t)$ is the time dependent fault content function and B is the constant FRF.

Solving the differential equations (3) and (4) using the change point concept, we get the mean value function

$$m(t) = \begin{cases} \frac{a}{(1-\beta_1)}\left(1 - e^{-B(1-\beta_1)b_1 t}\right), & 0 \leqslant t \leqslant \tau, \\ \frac{a}{(1-\beta_2)}\left(1 - e^{-B[(1-\beta_1)b_1\tau + (1-\beta_2)b_2(t-\tau)]}\right) & \\ \quad + \frac{(\beta_1-\beta_2)m(\tau)}{1-\beta_2}, & t > \tau. \end{cases} \tag{5}$$

The corresponding failure intensity function is

$$\lambda(t) = \frac{dm(t)}{dt}. \tag{6}$$

3 Estimation of parameters

The parameter estimation is of primary importance in software reliability prediction. The maximum likelihood estimation (MLE) is performed to evaluate the parameters for mean value function $m(t)$.

Once the analytical solution for $m(t)$ is known for a given model, the parameters involved in the solution need to be determined. We suggest the logarithmic MLE technique for estimating the unknown parameters. For n faults, let t_k, for $1 \leqslant k \leqslant n$, be the random time at which the kth fault occurs, with $0 < t_1 < t_2 < \cdots < t_k$. Then the MLE is

$$\begin{aligned} L &= \Pr[N(t_1) = m_1, N(t_2) = m_2, \ldots, N(t_n) = m_n] \\ &= \prod_{k=1}^{n} \frac{[m(t_k) - m(t_{k-1})]^{(m_k - m_{k-1})}}{(m_k - m_{k-1})!} \exp\left[m(t_k) - m(t_{k-1})\right], \end{aligned} \tag{7}$$

where $N(t_k) = m_k$ is the actual cumulative number of faults detected at time t_k. Applying a logarithm and taking partial derivatives of equation (7) with respect to some unknown parameter φ and setting those equations equal to zero, we obtain

$$0 = \sum_{k=1}^{n} \frac{\frac{\partial}{\partial\varphi} m(t_k) - \frac{\partial}{\partial\varphi} m(t_{k-1})}{m(t_k) - m(t_{k-1})} (m_k - m_{k-1}) - \frac{\partial}{\partial\varphi} m(t_n) \tag{8}$$

For our model φ takes the values a, b_1, b_2, β_1, β_2 and B.

By replacing φ by a, b_1, β_1 and B in equation (8) we get the system of nonlinear equations

$$a = \frac{(1-\beta)m_n}{1 - e^{x_n}}, \tag{9}$$

$$\sum_{k=1}^{n} \frac{(t_k e^{x_k} - t_{k-1} e^{x_{k-1}})(m_k - m_{k-1})}{e^{x_{k-1}} - e^{x_k}} = \frac{m_n t_n e^{x_n}}{1 - e^{x_n}}, \tag{10}$$

Table 1: Summary of data set.

Data set	Reference	Errors	Software project
DSI	Pham [13]	136	Bell Laboratories, 25 CPU hours execution time, 21 700 lines of code.

and

$$\sum_{k=1}^{n} \frac{[(1-Bb_1t_{k-1})e^{x_{k-1}} - (1-Bb_1t_k)e^{x_k}](m_k - m_{k-1})}{e^{x_{k-1}} - e^{x_k}} = \frac{a(1-e^{x_n})(1-Bb_1t_n)}{1-\beta_1}, \tag{11}$$

where $x_k = -B(1-\beta_1)b_1t_k$.

Now solve the above nonlinear system of equations using the Newton–Raphson method for finding unknown parameters a, b_1, β_1 and B, by using the data set given in Table 1.

4 Software release time based on reliability criteria

By using the mean value function we evaluate some useful metrics which are used to calculate software quality. There are two commonly accepted metrics, namely, reliability of the software and the mean time to failures (MTTF).

Generally, the software-release time problem is associated with the reliability of a software system. In this section, we discuss the software release policy based on the reliability criterion. If we know that the software has reached its supreme level of reliability for a particular time, then we ascertain the

right time to release the software. The conditional reliability function over some time period Δt is

$$R(\Delta t/t) = e^{m(t)-m(t+\Delta t)}\,. \tag{12}$$

MTTF is defined as the average elapsed time that passes before a failure occurs in a software system (cf. Lyu [7]). The cumulative and instantaneous MTTF are, respectively,

$$\mathrm{MTTF_c} = \frac{t}{m(t)} = \frac{t}{a(1-e^{-Bb(1-\beta)t})}, \tag{13}$$

$$\mathrm{MTTF_I} = \frac{1}{\lambda(t)}. \tag{14}$$

4.1 Software release time based on cost requirement

Let $C(T)$ be the cost function of a software at time T. Then,

$$C(T) = C_1 m(T) + C_2[m(\infty) - m(T)] + C_3 T\,, \quad T \geqslant 0\,, \tag{15}$$

where C_1 is the expected cost of removing a fault during the testing phase, C_2 is the expected cost of removing a fault during the operation phase with $C_2 > C_1$, and C_3 is the expected cost per unit time of testing. By taking the derivative of above equation with respect to T and equating to zero, an optimal release time T_c that minimizes the cost function is obtained.

4.2 Software release time based on reliability and cost requirement

The release time of a software is determined by considering both the reliability and cost requirements. With these, the optimization problem is [13]

$$\text{minimize} \quad C(T) \quad \text{subject to} \quad R(\Delta T \mid T) \geqslant R_0\,. \tag{16}$$

Let T_c be the solution of the minimum cost function $C(T_c)$ and T_r be the solution of $R(\Delta T \mid T) = R_0$. The optimal release time is

$$T^* = \max(T_c, T_r)\,. \tag{17}$$

5 Performance measures for goodness of fit

In this section we compare the performances of existing SRGMs and the proposed model with the help of the data set given in Table 1. The comparison criteria for our model is described below.

5.1 Mean square error

We validate the analytical results of our model by comparing the simulated fault data with the observed data. The difference between the simulated data $m(t_k)$ and the cumulative number of detected faults m_k is measured by the mean square error (cf. Lyu and Nikora [8])

$$\text{MSE} = \sum_{k=1}^{n} \frac{[m(t_k) - m_k]^2}{n}. \tag{18}$$

A smaller MSE indicates a smaller fitting error and gives a better goodness of fit.

5.2 Accuracy of estimation

The accuracy of estimation is (cf. Musa et al. [11])

$$\text{AE} = \left| \frac{m_k - a}{m_k} \right|, \tag{19}$$

where m_k is the actual cumulative number of detected faults after the test and a is the estimated number of faults.

Table 2: Comparison criteria.

Model	a	b_1	B	β_1	AE	MSE	MTTF
Goel & Okumoto [3]	142.3	0.124	–	–	0.04643	276.4	10.42
Hsu et al. [4]	136.1	0.299	0.459	–	0.00056	222.6	12.56
Proposed model	125.6	0.39	0.49	0.079	0.07631	140.9	20.81

Table 3: Optimal release values.

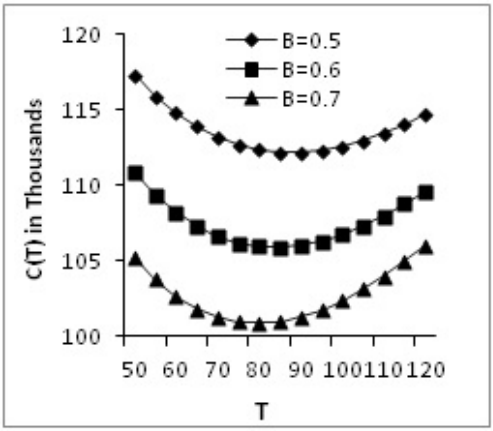
Model	T^* (hours)	$C(T^*)$ (dollars)
Goel & Okumoto [3]	26.42	56488
Hsu et al. [4]	24.80	53751
Proposed model	19.89	47917

6 Numerical results

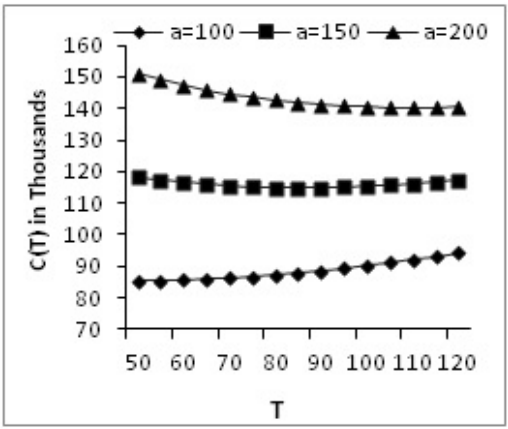
In this section we provide numerical results, obtained using Matlab, to examine the validity of the proposed model. The numerical results are presented to visualize the effects of different parameters such as FRF and cost of removing an error during the testing period. The estimation of parameters before the change point are calculated using the data set given in Table 1.

The parameter of estimation results and the comparison criteria are listed in Table 2. The optimal release time of the software which satisfies both cost and reliability requirements is displayed in Table 3. From Table 2 it is seen that our model provides more accurate MSE values than other models. We also observe that our model provides better optimal release times in comparison to other models.

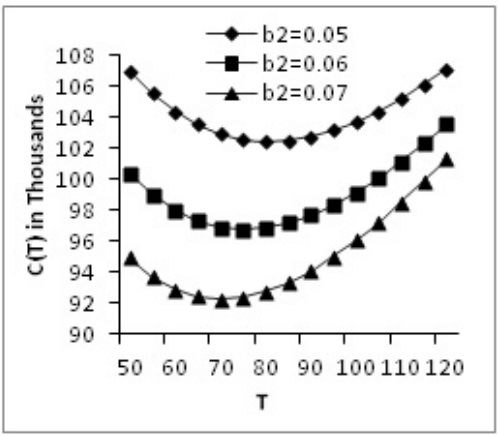
We also present the sensitivity analysis for a constant type FRF after the change point. The effect of various parameters on the total cost are explored by varying the parameters B (fault reduction factor), a (initial number of errors), b_1 and b_2 (error detection rates) and β_1 and β_2 (fault introduction rates). For computational purpose, the cost elements are fixed at $C_1 = \$300$,



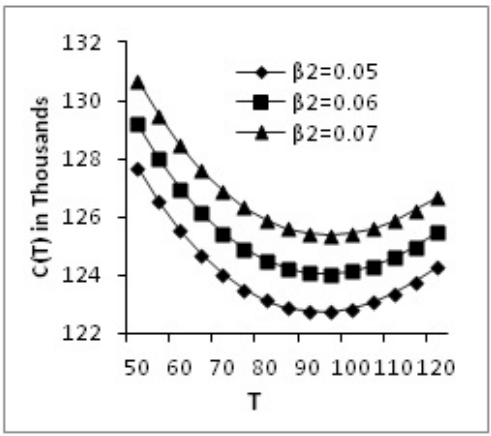
(i)



(ii)



(iii)



(iv)

Figure 1: Effect of various parameters on C(T) (in thousands of dollars) over T (in hours) after change point.

$C_2 = \$900$, $C_3 = \$400$. The effect of sensitive parameters on the total cost is depicted in Figure 1. In all plots $C(T)$ first decreases and then increases during the testing time. However, $C(T)$ decreases with increases in B and b_2. In contrast, $C(T)$ increases with increases in the number of estimated faults a and the fault introduction rate β_2.

7 Conclusion

We developed a SRGM by incorporating more novel features, namely, imperfect debugging and a change point. Our model is more realistic and suitable for modelling the real time software reliability growth. The software cost subject to the reliability constraint may provide an insight into achieving maximum reliability within a given budget. The proposed model and findings were validated via numerical illustrations, which demonstrated the applicability of investigations carried out for different types of software and large-scale real time embedded systems.

Acknowledgements The authors thank the unknown reviewers for helping to make a better quality article for publication. The second author is thankful to the MHRD, Government of India and CSIR for providing financial support to attend and present the article at the EMAC2013 conference.

References

[1] Ahmad, N., Khan, M. G. M. and Rafi, L. S. Analysis of an inflection S-shaped software reliability model considering log-logistic testing-effort and imperfect debugging, *Int. J. Comput. Sci. Net. Sec.*, 11(1)161–171, 2011. http://paper.ijcsns.org/07_book/201101/20110125.pdf. C184

[2] Chang, Y.-C. and Liu, C. A generalized JM model with applications to imperfect debugging in software reliability, *Appl. Math. Model.*, 33(9)3578–3588, 2009. doi:10.1016/j.apm.2008.11.018. C184

[3] Goel, A. L. and Okumoto, K. Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE T. Reliab.*, 28(3):206–211, 1979. doi:10.1109/TR.1979.5220566. C191

[4] Hsu, C.-J., Huang, C.-Y. and Chang, J.-R. Enhancing software reliability modelling and prediction through the introduction of time-variable fault reduction factor, *Appl. Math. Model.*, 35(1):506–521, 2011. doi:10.1016/j.apm.2010.07.017. C184, C185, C191

[5] Jha, P. C., Gupta, D., Yang, B. and Kapur, P. K. Optimal testing resource allocation during module testing considering cost, testing effort and reliability, *Comput. Int. Eng.*, 57(3):1122–1130, 2009. doi:10.1016/j.cie.2009.05.001. C185

[6] Kapur, P. K., Kumar, A., Yadav, K. and Khatri, S. K. Software reliability growth modelling for errors of different severity using change point, *Int. J. Qual., Reliab., Safe. Eng.*, 14(4):311–326, 2007. doi:10.1142/S0218539307002672. C184

[7] Lyu, M. R. Handbook of Software Reliability Engineering, McGraw-Hill, New York, 1996. C184, C189

[8] Lyu, M. R. and Nikora, A. Applying reliability models more effectively (software), *IEEE Software*, 9(4):43–52, 1992. doi:10.1109/52.143104. C190

[9] Musa, J. D. A theory of software reliability and its application, *IEEE T. Software Eng.*, 3(1):312–327, 1975. doi:10.1109/TSE.1975.6312856. C184

[10] Musa, J. D. The measurement and management of software reliability, *P. IEEE*, 68(9):1131–1143, 1980. doi:10.1109/PROC.1980.11812. C184

[11] Musa, J. D., Lamino, A. and Okumoto, K. Software reliability: Measurement, Prediction, Application, McGraw-Hill, New York, 1987. C184, C190

[12] Okumoto, K. and Goel, A. L. Optimum release time for software system based on reliability and cost criteria, *J. Syst. Software*, 1:315–318, 1980. doi:10.1016/0164-1212(79)90033-5. C185

[13] Pham, H. Software Reliability, Springer-Verlag, Singapore, 2000. C184, C188, C189

[14] Quadri, S. M. K. and Ahmad, N. Software reliability growth modelling with new modified Weibull testing- effort and optimal release policy, *Int. J. Comput. Appl.*, 6(12), 2010. doi:10.5120/1127-1477. C185

[15] Quadri, S. M. K., Ahmad, N. and Farooq, S. U. Software reliability growth modelling with generalized exponential testing-effort and optimal software release policy, *Global J. Comput. Sci. Tech.*, 11(2):27–42, 2011. http://computerresearch.org/stpr/index.php/gjcst/article/view/605. C185

[16] Shyur, H.-J. A stochastic software reliability model with imperfect debugging and change-point, *J. Syst. Software*, 66(2):135–141, 2003. doi:10.1016/S0164-1212(02)00071-7. C184

[17] Xie, M. Software Reliability Modelling, World Scientist, Singapore, 1991. C184

[18] Xie, M. and Yang, B. A study of the effect of imperfect debugging on software development cost model, *IEEE T. Software Eng.*, 29(5):471–473, 2003. doi:10.1109/TSE.2003.1199075. C184

[19] Li, X., Xie. M. and Szu H. N. Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points, *Appl. Math. Model.*, 34(11):3560–3570, 2010. doi:10.1016/j.apm.2010.03.006. C184

[20] Zhao, J., Liu, H.-W., Cui, G. and Yang, X.-Z. Software reliability growth model with change-point and environmental function, *J. Syst. Software*, 79(11):1578–1587, 2006. doi:10.1016/j.jss.2006.02.030. C184

Author addresses

1. **Madhu Jain**, Department of Mathematics, Indian Institute of Technology Roorkee, India
mailto:drmadhujain.iitr@gmail.com

2. **T. Manjula**, Department of Mathematics, Indian Institute of Technology Roorkee, India
mailto:manju.iitr09@gmail.com

3. **T. R. Gulati**, Department of Mathematics, Indian Institute of Technology Roorkee, India
mailto:trgulati@gmail.com