ISSN 0974 - 9373

Vol. 14 No.3 (2010) **Journal of International Academy of Physical Sciences** pp. 345-355

Warranty Cost Analysis of Software Reliability Growth Model (SRGM) with Imperfect Debugging and Change-point*

M. Jain
Department of Mathematics, Indian Institute of Technology, Roorkee-247667 (India)
E-mail- madhufma@iitr.ernet.in , madhujain@sancharnet.in

S. C. Agrawal and Priyanka Agarwal
School of Basic and Applied Sciences, Shobhit University, Meerut-250001 (India)
E-mail- scagrawal7@rediffmail.com , priyanka1354@gmail.com

(Received December 21, 2009)

Abstract: In this paper, the optimal release policy for constructing the software reliability growth model (SRGM) that incorporates both imperfect debugging and change-point concepts has been investigated. In any realistic situations the change-point problem may be realized due to change in many factors like testing strategies, environment, resource allocation, etc.. It is important to ensure when to stop testing or when to release the software so that the total system development cost can be minimized subject to reliability constraint. To evaluate the total expected cost, a warranty cost model using non-homogeneous process along with change-point phenomenon is discussed. Optimal release policies based on cost and reliability criteria are constructed for determining fairly accurate optimal software release time. The phenomenon and applicability of the proposed model are established via numerical results.

Keywords: Software reliability growth model (SRGM), Non-homogeneous Poisson process (NHPP), Change-point, Warranty cost, Optimal release time, Reliability.

1. Introduction

It is very important for software developer to control a software development process in terms of cost, reliability and optimal release time. Software is released in the market for operational use when it is tested perfectly. But a major problem for software developers is to decide at what time, the testing should be stopped and software should be released so that the total cost can be minimized subject to reliability constraint. In literature,

*Presented at CONIAPS XI, University of Allahabad, Feb. 20-22, 2010.

some research papers on optimization problem have also appeared. Yamada[1] worked on the policies with warranty period and maintenance cost model. Kimura et al.[2] illustrated the economic analysis of software release problem with warranty cost and reliability requirement. Zheng[3] and Jain and Priya[4] obtained release policies with reliability constraint for software testing time. Huang et al.[5] analyzed the optimal release time for software systems considering cost, testing effort and test efficiency. The optimal testing resource allocations for modular software system considering cost and reliability constraints are investigated by Jha et al.[6]**.** One of the key issues of software engineering is to manage the process of testing under a limited budget in a specified time interval. In this paper, we investigate how to integrate change-point concept into imperfect debugging SRGM with warranty cost. We suggest the optimal release policies under cost and reliability constraints.

2. SRGM with Imperfect Debugging and Change-Point

Most of the software reliability growth models focus on the software testing phase, where software defects are detected, isolated and removed and then software tends to grow. In this section, a SRGM is developed which incorporates both imperfect debugging and change-point concepts. It is assumed that during the software testing, fault detection rate and fault introduction rate change at some instant τ.

The fault detection rate function with change-point is defined as

$$b(t)=\begin{cases} b_1, & \text{for } 0 \le t \le \tau \\ b_2, & \text{for } \quad t > \tau \end{cases} \tag{1}$$

The fault introduction rate during testing with change-point is defined as

$$\beta(t)=\begin{cases} \beta_1, & \text{for } 0 \le t \le \tau \\ \beta_2, & \text{for } \quad t > \tau \end{cases} \tag{2}$$

The proposed model is based on the following assumptions:

- The fault removal phenomenon is modeled by non-homogeneous Poisson process (NHPP).
- The software system is subject to failures at random times caused by the manifestation of remaining faults in the system.
- The proportionality of fault detection is constant over time.
- The time between $(i-1)^{th}$ and i^{th} failures depends on the time to the $(i-1)^{th}$ failure.

- On the detection/removal of a software failure, it may be possible that the effort to remove the failure may not be perfect i.e. due to imperfect debugging a new fault may be introduced during removal of the error.
- The total number of faults at the beginning of the testing phase is finite i.e. the imperfect error debugging does not increase the initial error content.

These are the notations which have been used to formulate the model:

m(t)	:	Mean value function in the NHPP model.
$\lambda(t)$	:	Failure intensity function.
a	:	Initial number of errors in the software before starting of testing phase.
b	:	Fault detection rate.
β	:	Fault introduction rate.
C_0	:	Initial testing cost.
C_t	:	Testing cost per unit time.
C_w	:	Maintenance cost per fault during the warranty period.
T	:	Release time of the software.
T*	:	Optimal release time of the software.
T_w	:	Warranty period.
α	:	Discount rate of the cost.
EC(T)	:	Expected total maintenance cost of the software.
$C_w(T)$	:	Maintenance cost during the warranty period.

The mean value function m(t) can be obtained as [cf. Shyur, 2003]:

$$(3)\qquad m(t)=\begin{cases}\dfrac{a}{1-\beta_1}\left[1-\exp\left(-(1-\beta_1)b_1 t\right)\right], & 0<t<\tau \\ \dfrac{a}{1-\beta_2}\left[1-\exp\left(-(1-\beta_1)b_1\tau-(1-\beta_2)b_2(t-\tau)\right)\right]+\dfrac{m(\tau)(\beta_1-\beta_2)}{1-\beta_2}, & t>\tau\end{cases}$$

The corresponding failure intensity function is

$$(4)\qquad \lambda(t)=\frac{dm(t)}{dt}=\begin{cases}ab_1\left[\exp\left(-(1-\beta_1)\,b_1 t\right)\right], & 0<t<\tau \\ ab_2\left[\exp\left(-(1-\beta_1)\,b_1\tau-(1-\beta_2)\,b_2(t-\tau)\right)\right], & t>\tau\end{cases}$$

3. Warranty Cost Model

The total expected software maintenance cost is given by

$$EC(T) = C_0 + C_t \int_0^T \exp(-\alpha T)dt + C_w(T) \tag{5}$$

There are two cases for maintenance cost during warranty period:

Case I: In this case, it is assumed that during the warranty period, the software reliability growth does not occur. Here we assume that only minor errors that will not affect the reliability of the software, can be corrected. Then maintenance cost during warranty is obtained as

$$C_w(T) = \begin{cases} C_w \int_T^{T+T_w} \lambda(T)\exp(-\alpha t)\,dt, & 0<t<\tau \\ C_w \int_T^{T+T_w} \lambda(T)\exp(-\alpha t)\,dt, & t>\tau \end{cases} \tag{6}$$

Now, using equations (5)-(6), we get

$$EC(T) = C_0 + C_t \int_0^T \exp(-\alpha t)\,dt + C_w \frac{ab_1}{\alpha}\exp\left(-(1-\beta_1)b_1 T\right)\exp(-\alpha T)\left[1-\exp(-\alpha T_w)\right], \quad \text{for } 0<t<\tau \tag{7}$$

$$EC(T) = C_0 + C_t \int_0^T \exp(-\alpha t)\,dt + C_w \frac{ab_2}{\alpha}\exp\left(-(1-\beta_1)b_1\tau-(1-\beta_2)b_2(T-\tau)\right)\exp(-\alpha T)\left[1-\exp(-\alpha T_w)\right], \quad \text{for } t>\tau \tag{8}$$

Differentiating equation (7) with respect to 'T' and equating to zero, where $(0 \le t \le \tau)$, we get

$$T = T_1 = \frac{1}{b_1(1-\beta_1)}\ln\frac{ab_1}{\alpha}\frac{C_w}{C_t}\left[\alpha+b_1(1-\beta_1)\right]\left[1-\exp(-\alpha T_w)\right] \tag{9}$$

and differentiating equation (9) with respect to 'T' and equating to zero, where $(t > \tau)$, we have

$$T = T_2 = \frac{1}{b_2(1-\beta_2)}\ln\frac{ab_2}{\alpha}\frac{C_w}{C_t}\exp(-\tau[(1-\beta_1)b_1 - (1-\beta_2)b_2])\left[\alpha+b_2(1-\beta_2)\right]\left[1-\exp(-\alpha T_w)\right] \tag{10}$$

Since $\left[\frac{d^2EC(T)}{dT^2}\right]_{T=T_1} > 0$, and $\left[\frac{d^2EC(T)}{dT^2}\right]_{T=T_2} > 0$,

EC(T) has a minimum value at $T_1 = T^*,\ 0 < t \le \tau$; and $T_2 = T^*,\ t > \tau$.

Case II: Here we assume that during the warranty period, the software reliability growth occurs, even after the testing phase. In this case only major errors that will improve the reliability of the software can be corrected. Then $C_W(T)$ is given by

$$C_w(T) = \begin{cases} C_w \int_T^{T+T_w} \lambda(t)\exp(-\alpha t)dt, & \text{for } 0 \le t \le \tau \\ C_w \int_T^{T+T_w} \lambda(t)\exp(-\alpha t)dt, & \text{for } t > \tau \end{cases} \tag{11}$$

Substituting above values from eq. (11) in eq. (6), we get

$$EC(T) = C_0 + C_t \int_0^T \exp(-\alpha t)\,dt + C_w \frac{ab_1}{\left[(1-\beta_1)b_1+\alpha\right]} \exp\left(-(1-\beta_1)b_1+\alpha\right)T \left[1-\exp\left(-(1-\beta_1)b_1+\alpha\right)T_w\right], \quad \text{for } 0 < t < \tau \tag{12}$$

$$EC(T) = C_0 + C_t \int_0^T \exp(-\alpha t)\,dt + C_w \frac{ab_2}{\left[(1-\beta_2)b_2+\alpha\right]} \exp\left[z-\left((1-\beta_2)b_2+\alpha\right)T\right] \left[1-\exp\left[-\left((1-\beta_2)b_2+\alpha\right)T_w\right]\right] \quad \text{for } t > \tau \tag{13}$$

Differentiating equation (11) with respect to 'T' and equating to zero, where $(0 \le t \le \tau)$, we get

$$T = T_3 = \frac{1}{b_1(1-\beta_1)} \ln \frac{C_w}{C_t} ab_1\left[1-\exp\left[-\left((1-\beta_1)b_1+\alpha\right)T_w\right]\right] \tag{14}$$

Again, differentiating equation (12) with respect to 'T' and equating to zero, where $(t > \tau)$, we find

$$T = T_4 = \frac{1}{b_2(1-\beta_2)} \ln \frac{C_w}{C_t} ab_2 \exp(z)\left[1-\exp\left[-\left((1-\beta_1)b_1+\alpha\right)T_w\right]\right] \tag{15}$$

where $z = -(1-\beta_1)b_1\tau + (1-\beta_2)b_2\tau$.

Again $\left[\frac{d^2EC(T)}{dT^2}\right]_{T=T_3} > 0$, and $\left[\frac{d^2EC(T)}{dT^2}\right]_{T=T_4} > 0.$

Thus EC(T) has a minimum value at $T_3 = T^*$ for $0 \le t \le \tau$, and $T_4 = T^*$ for $t > \tau$.

4. Warranty Cost Model with Reliability Constraint

The software reliability of the NHPP model is defined as the probability that a software failure will not occur during the testing time interval (T, T+x]. The software reliability function is formulated as:

$$R\left(x/T\right)=\exp\left[-\left\{m(T+x)-m(T)\right\}\right]$$

$$=\begin{cases}\exp\left[-\exp\left(-(1-\beta_1)b_1T\right).m_1(x)\right], & \text{for } 0<t<\tau\\ \exp\left[-\exp\left(-(1-\beta_1)b_1\tau-(1-\beta_2)b_2(T-\tau)\right).m_2(x)\right], & \text{for } t>\tau\end{cases} \quad (16)$$

where $m_i(x)=\dfrac{a}{1-\beta_i}\left[1-\exp\left(-(1-\beta_i)b_i x\right)\right]$, $i = 1,2.$

Let T_{R1} be the optimal release time for the case (i) for testing time T_1 ($0 \le t \le \tau$) and T_2 ($t > \tau$) satisfying the relation $R\left(x/T_1\right)=R_0$, $R\left(x/T_2\right)=R_0$, respectively. Similarly T_{R2} be the optimal release time for the case (ii) for testing time T_3 ($0 \le t \le \tau$) and T_4 ($t > \tau$) satisfying the relation $R\left(x/T_3\right)=R_0$ and $R\left(x/T_4\right)=R_0$ respectively. Now, we get

$$T_{R1}=\frac{1}{b_1(1-\beta_1)}\left\{\ln\left(m_1(x)\right)-\ln\ln\left(\frac{1}{R_0}\right)\right\} \quad (17)$$

and

$$T_{R2}=\frac{1}{b_2(1-\beta_2)}\left\{\ln\left(m_2(x)\right)+\left((1-\beta_2)b_2-(1-\beta_1)b_1\right)\tau-\ln\ln\left(\frac{1}{R_0}\right)\right\} \quad (18)$$

Let R_0 ($0 < R_0 \le 1$) be the desired level of reliability and T = T* (optimum release time) which minimizes the expected cost of the software with reliability objective R_0. Thus, the optimization problem can be formulated as

$$\text{Minimize EC(T),}$$
$$\text{subject to } R(x/T) \ge R_0 \quad (19)$$

5. Optimal Software Release Policies

The optimal software release policies for the NHPP SRGM based on cost and reliability criteria are as follows:

Optimal Testing Time with Cost

(A) Optimal Testing Policy 1 (OTP 1) ***for case I:***

$P_{A.1}$: $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ and $\lambda(\tau) < \lambda(T_2)$, then $T^* = T_1$.

$P_{A.2}$: $\lambda(0) < \lambda(T_1) \leq \lambda(\tau)$ and $\lambda(\tau) > \lambda(T_2)$, then $T^* = T_2$.

$P_{A.3}$: $\lambda(0) < \lambda(T_1) \leq \lambda(\tau)$ and $\lambda(\tau) < \lambda(T_2)$, then $T^* = 0$.

$P_{A.4}$: $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ and $\lambda(\tau) > \lambda(T_2)$, then $T^* = \max(T_1, T_2)$.

(B) Optimal Testing Policy 2 (OTP 2) ***for case (ii):***

$P_{B.1}$: $\lambda(0) > \lambda(T_3) \geq \lambda(\tau)$ and $\lambda(\tau) < \lambda(T_4)$, then $T^* = T_3$.

$P_{B.2}$: $\lambda(0) < \lambda(T_3) \leq \lambda(\tau)$ and $\lambda(\tau) > \lambda(T_4)$, then $T^* = T_4$.

$P_{B.3}$: $\lambda(0) < \lambda(T_3) \leq \lambda(\tau)$ and $\lambda(\tau) < \lambda(T_4)$, then $T^* = 0$.

$P_{B.4}$: $\lambda(0) > \lambda(T_3) \geq \lambda(\tau)$ nd $\lambda(\tau) > \lambda(T_4)$, then $T^* = \max(T_3, T_4)$.

Optimal Testing Time with Cost and Reliability Constraints

(C) Optimal Testing Policy 3 (OTP 3) ***for case I:***

$P_{C.1}$: If $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ and $\lambda(\tau) < \lambda(T_2)$ and $R(x/0) > R_0$, then $T^* = T_1$.

$P_{C.2}$: If $\lambda(0) < \lambda(T_1) \leq \lambda(\tau)$ and $\lambda(\tau) > \lambda(T_2)$ and $R(x/0) > R_0$, then $T^* = T_2$.

$P_{C.3}$: If $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ and $\lambda(\tau) > \lambda(T_2)$ and $R(x/0) > R_0$, then $T^* = \max\{T_1, T_2\}$.

$P_{C.4}$: If $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ and $\lambda(\tau) < \lambda(T_2)$ and $R(x/0) > R_0$, then $T^* = 0$.

$P_{C.5}$: If $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ and $\lambda(\tau) < \lambda(T_2)$ and $R(x/0) < R_0$, then $T^* = \max\{T_1, T_{R1}\}$.

$P_{C.6}$: If $\lambda(0) < \lambda(T_1) \leq \lambda(\tau)$ and $\lambda(\tau) > \lambda(T_2)$ and $R(x/0) > R_0$, then $T^* = \max\{T_2, T_{R2}\}$.

$P_{C.7}$: If $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ and $\lambda(\tau) > \lambda(T_2)$ and $R(x/0) < R_0$, then $T^* = \max\{T_1, T_2, T_{R1}\}$.

(D) Optimal Testing Policy 4 (OTP 4) ***for case II:***

$P_{D.1}$: If $\lambda(0) > \lambda(T_3) \geq \lambda(\tau)$ and $\lambda(\tau) < \lambda(T_4)$ and $R(x/0) > R_0$, then $T^* = T_3$.

$P_{D.2}$: If $\lambda(0) < \lambda(T_3) \leq \lambda(\tau)$ and $\lambda(\tau) > \lambda(T_4)$ and $R(x/0) > R_0$, then $T^* = T_4$.

$P_{D.3}$: If $\lambda(0) > \lambda(T_3) \geq \lambda(\tau)$ and $\lambda(\tau) > \lambda(T_4)$ and $R(x/0) > R_0$, then $T^* = \max\{T_3, T_4\}$.

$P_{D.4}$: If $\lambda(0) > \lambda(T_3) \geq \lambda(\tau)$ and $\lambda(\tau) < \lambda(T_4)$ and $R(x/0) > R_0$, then $T^* = 0$.

$P_{D.5}$: If $\lambda(0) > \lambda(T_3) \geq \lambda(\tau)$ and $\lambda(\tau) < \lambda(T_4)$ and $R(x/0) < R_0$, then $T^* = \max\{T_3, T_{R1}\}$.

$P_{D.6}$: If $\lambda(0) < \lambda(T_3) \leq \lambda(\tau)$ and $\lambda(\tau) > \lambda(T_4)$ and $R(x/0) > R_0$, then $T^* = \max\{T_4, T_{R2}\}$.

$P_{D.7}$: If $\lambda(0) > \lambda(T_3) \geq \lambda(\tau)$ and $\lambda(\tau) > \lambda(T_4)$ and $R(x/0) < R_0$, then $T^* = \max\{T_3, T_4, T_{R1}\}$.

6. Sensitivity Analysis

To verify the proposed software reliability growth model that incorporates both imperfect debugging and change-point concept, we perform the sensitivity analysis by computing the expected maintenance cost and software reliability. Some default parameters are fixed as $C_0 = 100$, $C_t = 5$, $C_w = 150$, $T_w = 5$, $\beta = 0.04$, $x = 0.9$.

The graphical representation of the expected maintenance cost EC(T) has been done in figs 1- 2. The reliability R(T) has also been displayed in figs 3-5.

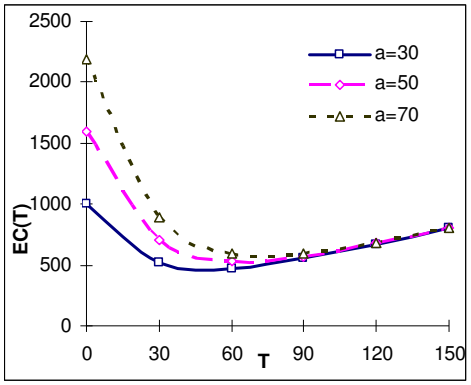
Figs 1(i)-1(iii) exhibit the effects of a, b_1 and α on EC(T) for the cases 1 whereas figs 2(i)-2(iii) depict the effects of same parameters for case 2 before the change-point. Figs 1(i) and 1(ii) reveal that EC(T) first decreases (approximately up to t = 40) and then after it increases for the testing time and finally it attains constant behaviour. We notice the same trend in figs 2(i) and 2(ii) which have been drawn for case 2 before change-point. Figs 1(ii) and 2(ii) also show the same pattern for b_1 with respect to time for case 1 and case 2, respectively. We also notice that EC(T) increases with respect to a and b_1 for both cases. In figs 1(iii) and 2(iii), EC(T) first decreases gradually up to t = 60 and after that it increases sharply. It is also seen that after t = 60, EC(T) decreases with respect to α.

Figs 3-5 illustrate the pattern of the reliability of the software by varying the parameters a, b_1, b_2 and τ for both cases i.e. before change-point and after change-point. Figs 5(i) and 6(i) show the trend of reliability by varying the parameter a before change-point and after change-point, respectively. We notice that after the change-point, the reliability increases as compare to before change-point. We also see the same pattern of reliability in figs 3(ii) and 4(ii). It can be easily seen that as initial number of faults (a) increases,

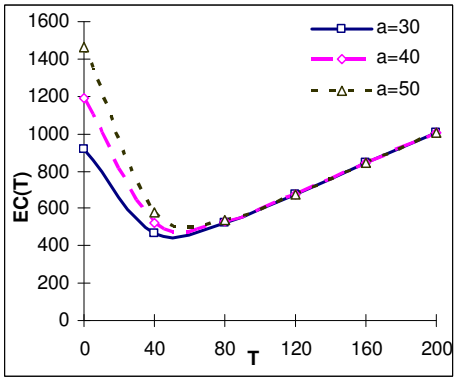
the reliability increases. We notice the same effect of parameters b_2 and τ on EC(T) in figs 5(i) and 5(ii) and in figs 3 (ii) and 4 (ii). Also, we found that EC(T) increases as error detection rate increases. In figs 5(i) and 5(ii), reliability is plotted by varying the parameters b_2 and τ, respectively, for after the change-point. In these graphs, the reliability is increasing sharply as compared to before change-point case.

4. Concluding Remarks

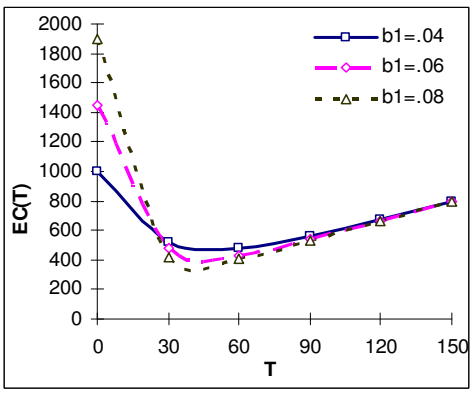
The software reliability growth model developed in this investigation includes the imperfect debugging phenomenon as well as change-point concepts. The software cost subject to reliability constraint discussed may provide an insight to achieve maximum reliability within a given budget. It is noticed that if the cost could be less than the given budget and the customers are satisfied, then it is a great profit for an organization. The proposed model and findings have been validated by taking numerical illustration, which demonstrates the applications of investigation done for different types of software and large-scale real time embedded systems.



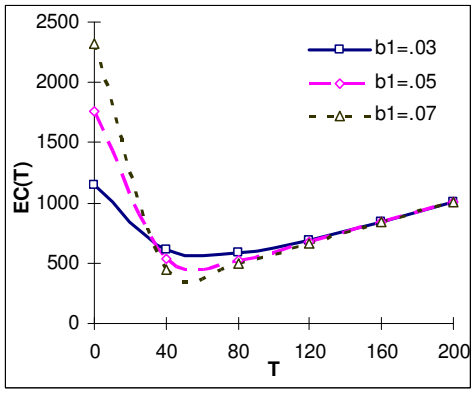
(i)



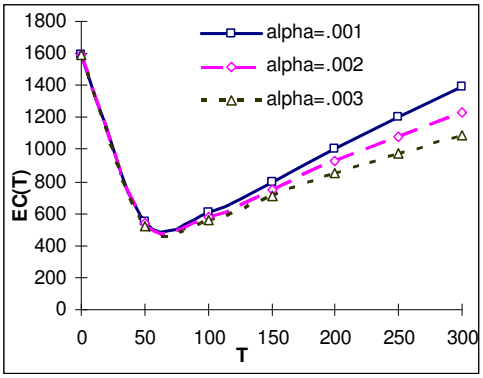
(i)



(ii)

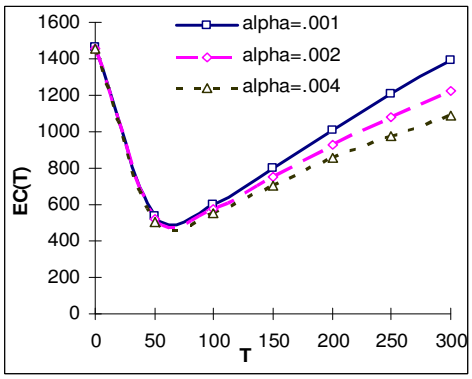


(ii)



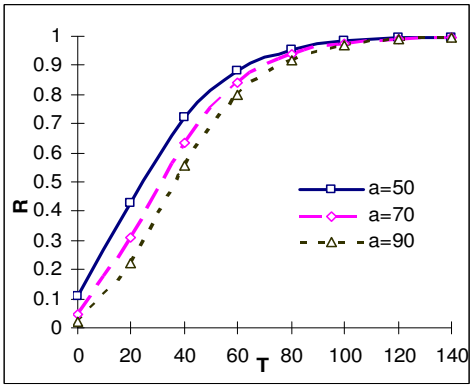
(iii)

Fig. 1: Expected maintenance cost vs T for case 1 for different values of (i) a (ii) b_1 (iii) α.

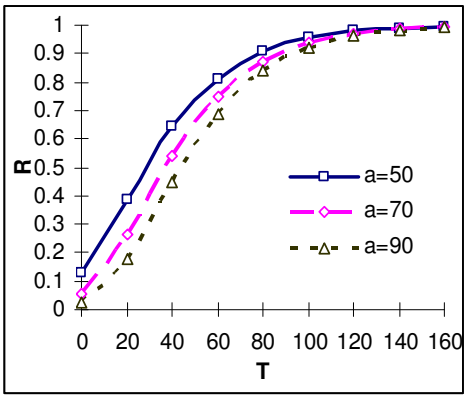


(iii)

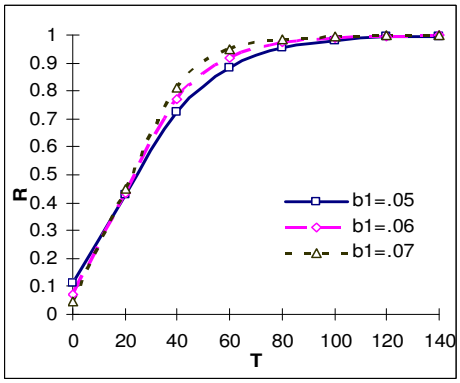
Fig. 2: Expected maintenance cost vs T for case 2 for different values of (i) a (ii) b_1 (iii) α.



(i)

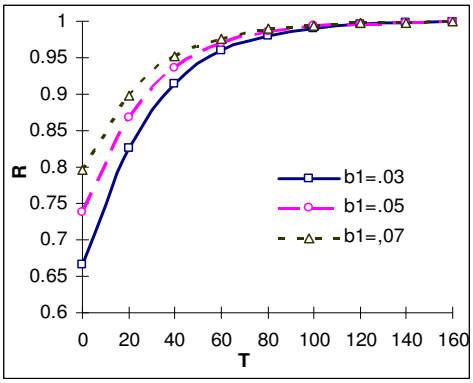


(i)



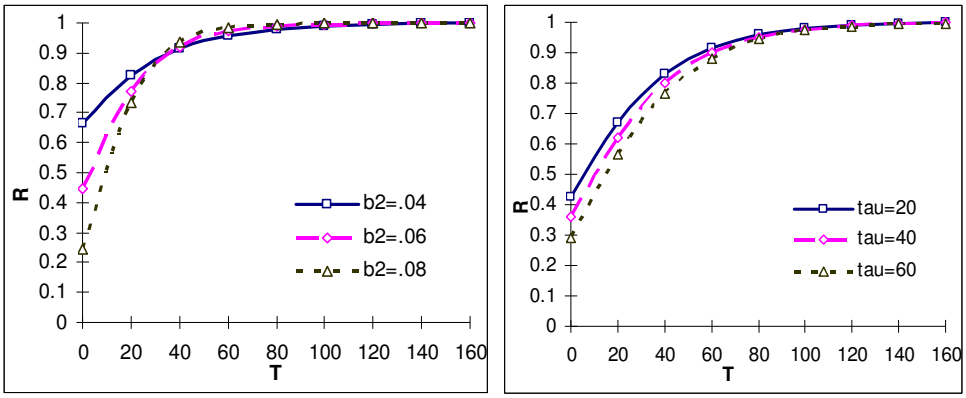
(ii)

Fig. 5: Software reliability vs T before change point for different values of (i) a (ii) b_1.



(ii)

Fig. 6: Software reliability vs T after change point for different values of (i) a (ii) b_1.



(i) **(ii)**

Fig. 7: Software reliability vs T after change-point for different values of (i) b_2 (ii) τ.

References

1. S. Yamada, Optimal release problems with warranty period based on a software maintenance cost model, *Transactions on IPS Japan*, **35-10** (1994) 2197-2202.
2. M. Kimura, T. Toyota and S. Yamada, Economic analysis of software release problems with warranty cost and reliability requirement, *Reliability Engineering and System Safety*, **66** (1999) 49-55.
3. S. Zheng, Dynamic release policies for software systems with a reliability constraint, *IIE Transactions*, **34-3** (2002) 253-262.
4. M. Jain and K. Priya, Optimal policies for software testing time, *Journal of the CSI*, **32- 3** (2002) 25-30.
5. C. Y. Huang and M. R. Lyu, Optimal release time for software systems considering cost, testing effort and test efficiency, *IEEE Transaction on Reliability*, **54** (2005) 583-591.
6. P. C. Jha, D. Gupta, B. Yang and P. K. Kapur, Optimal testing resource allocation during module testing considering cost, testing effort and reliability, *Computers and Industrial Engineering*, **57** (2009) 1122-1130.