Optimal Release times of a Software Cost Model with Consideration of Various Costs

Chong Hyung Lee
Division of Information Technology, Konyang University, Nonsan 320-711, KOREA
Kyu Beom Jang
Department of Statistics, Hallym University, Chunchon 200-702, KOREA
Dong Ho Park
Department of Statistics, Hallym University, Chunchon 200-702, KOREA

Abstract

Software system which is essential in operating the computer has gradually become an indispensable element in many aspects of our daily lives and an important factor in numerous systems. In recent years, software cost sometimes exceeds the cost of maintaining the hardware system. In addition to the cost necessary to develop the new software system and to maintain the system, the penalty costs incurred due to software failures are even more significant. In this paper, a cost model incorporating the warranty cost, debugging costto remove each fault detected in the software system, and delivery delay cost is developed. A software reliability model based on non-homogeneous Poisson process(NHPP) is established and the optimal software release policies to minimize the expected total software cost are discussed. Numerical examples are provided to illustrate the results.

1. Introduction

The breakdown of a computer system, caused by software faults, may result in serious damages in so many situations. Thus, the improvement of software reliability is one of the key issues in developing the modern software products. In fact, the software reliability is an important characteristic measuring the quality of software as a taken-for-granted quality. It is defined as the probability of no occurrence of software failures during a certain period under some specified conditions. A software failure is defined as an unacceptable departure from normal program operation cased by a software fault latent in the software. Although the software system is debugged during its testing phase before it is released to the user, it is in general impossible to release a fault-free software product. Thus, the problem of determining the best possible software release time with respect to its cost, reliability and other relevant factors is quite critical to the software developer. If the testing period is too short, the software development cost is saved, but the software reliability may suffer while being operated by the users and the repair costs that is incurred by the software failure may be increasing. On the other hand, if the testing period is too long, then the software reliability can be improved, but the debugging cost of the software also increases. Therefore, various optimal software release policies which determine the best possible release time satisfying several desirable requirements have been developed and proposed in the literature.

Goel and Okumoto (1979) proposed a software reliability growth model based on

NHPP and Yamada, Tokuno and Osaki (1992) and Pham (1996) developed several types of NHPP models which assume the imperfect debugging environments. By incorporating the software cost structure into the software reliability growth model, Yamada and Osaki (1985) discussed the optimal software release policies which minimize the software operating cost under the constraints with regard to a software reliability requirement. Yun and Bai (1990), Pham (1996) and Lee, Nam and Park (2002) also considered the situation when the software life cycle is a unknown constant and derived a cost model assuming a random life cycle as well as the penalty cost model with consideration of the delayed delivery of the software system.

In this paper, we present the mathematical expressions to evaluate the expected cost of operating the software based on NHPP model. The proposed cost model herein includes the cost to carry out testing, the cost of removing each fault detected in the software system, and the delivery delay cost. Section 2 gives the notations and necessary assumptions. In Section 3, the software cost model including the warranty cost and delivery delay cost is proposed and the optimal software release policy is discussed in Section 4. Numerical results are presented for illustrative purposes in Section 5.

2. Notations and assumptions

2.1 Notations

c_1	software test cost per unit time
$c_2(c_3)$	cost of fault removal per unit time during testing(warranty) period.
$N(t)$	number of faults detected up to time t, $m(t)=E[N(t)]$
t^*	optimal testing period.
Y_i	random variable representing time to remove the ith fault during testing period.
W_i	random variable representing time to remove the ith fault during warranty period.
t_W	length of warranty period
T_S	random variable representing the scheduled software delivery time
$c_p(t)$	cost incurred due to the delivery delay by t
$C_j(t)$	cost of type j, j = 1,2,3,4, $C(t)=\sum_{j=1}^{4} C_j(t)$

2.2 Assumptions

1. The cost of fault removals during the testing period is proportional to the total time needed to remove all the faults detected during this period.
2. The cost of fault removals during the warranty period is proportional to the total time needed to remove all the faults detected in the time interval $[t, t+t_W]$.
3. It takes a certain length of time to remove each fault during the testing and warranty period and it is assumed that the time needed to remove each fault follows a right-truncated exponential distribution and does not exceed the maximum time to remove a fault.
4. If the testing takes longer than T_S, which is set to meet the delivery schedule of a software system, an additional cost is incurred for the delay until the actual software release time t. It is assumed that T_S has a cumulative distribution function $G(t)$ with $G(0)=0$ and $\lim_{t\to\infty} G(t)=1$.

3. Software cost model

Under the assumptions given in Section 2.2, it is possible to establish a software cost model in terms of the following costs: testing cost, fault removal costs during the testing and warranty period, delivery delay cost, and our model is based on Goel-Okumoto NHPP model. The Goel-Okumoto model assumes that the number of faults detected at time t is proportional to the number of faults remaining at that time in the software. This means that the probability of a software failure due to fault is constant. The expected number of faults to be detected by time t, $m(t)$, can be obtained as follows:

$$m(t) = a(1 - e^{-bt}),$$

where a is the expected total number of faults that exist in a software before testing and b is the fault detection rate per fault. The corresponding four expected costs can be evaluated as follows:

1. Cost for testing

$$EC_1(t) = c_1 t$$

2. Cost of fault removal during testing period

$$\begin{aligned} EC_2(t) &= E\left[c_2 \cdot \sum_{i=1}^{N(t)} Y_i\right] \\ &= c_2 \cdot E[N(t)] \cdot E(Y_i)] \\ &= c_2 \cdot m(t) \cdot \mu_Y \end{aligned}$$

where μ_Y, which denotes the expectation of random variable Y_i, is the expected time needed to remove a fault during the testing period.

3. Cost of fault removal during warranty period,

$$\begin{aligned} EC_3(t) &= E\left[c_3 \cdot \sum_{i=N(t)}^{N(t+t_W)} W_i\right] \\ &= c_3 \cdot E[N(t+t_W) - N(t)] \cdot E(W_i)] \\ &= c_3 \cdot [m(t+t_W) - m(t)] \cdot \mu_W \end{aligned}$$

where μ_W, which denotes the expectation of random variable W_i , is the expected time needed to remove a fault during the warranty period.

4. Cost for delayed delivery during $[T_S, t]$

$$EC_4(t) = \int_0^t c_p(t-\tau)dG(\tau)$$

It is assumed that the scheduled software delivery time T_S has an exponential distribution with a finite mean λ_S. In this paper, we consider two special types of delivery delay cost: $c_p(t) = k_1 t$, $k_1 > 0$, linearly increasing function and $c_p(t) = k_2 \cdot [\exp(\gamma t - 1)]$, $\gamma > 0$, $k_2 > 0$, exponentially increasing function. Suppose that the random scheduled delivery time T_S is deterministic to be equal to t_S, then $G(t)$ is defined as

$$G(t) = \begin{cases} 1, & t \geq t_S \\ 0, & t < t_S, \end{cases}$$

where t_S is a constant.

Therefore, adding all four relevant costs, the expected total software cost $EC(t)$ can be expressed as

$$\begin{aligned} EC(t) &= EC_1(t) + EC_2(t) + EC_3(t) + EC_4(t) \\ &= c_1 t + c_2\mu_Y m(t) \\ &+ c_3\mu_W[m(t+t_W) - m(t)] \\ &+ \int_0^t c_p(t-\tau)dG(\tau) \end{aligned} \quad (1)$$

4. Optimal software release polices

In this section, we study the behavior of the expected total software cost $EC(t)$, given in (1), to determine the optimal software release time, t^*, which minimizes such an expected cost. For this purpose, we evaluate the first and second order derivatives of the equation (1) with respect to t to obtain the following expressions. Denote

$$\begin{aligned} y(t) &= d[EC(t)]/dt \\ &= c_1 + c_2\mu_Y ab \cdot \exp(-bt) \\ &\quad + c_3\mu_W ab[\exp(-bt_W) - 1] \cdot \exp(-bt) \\ &\quad + d/dt \int_0^t c_p(t-\tau)dG(\tau), \end{aligned}$$

and

$$\begin{aligned} w(t) &= d^2[EC(t)]/dt^2 \\ &= \exp(-bt) \cdot [u(t) - C] , \end{aligned} \quad (2)$$

where

$$u(t) = \exp(bt) \cdot d^2/dt^2 \int_0^t c_p(t-\tau)dG(\tau)$$

and

$$C = ab^2 c_2\mu_Y + ab^2 c_3\mu_W[\exp(-bt_W) - 1] .$$

As given in Sections 4.1 and 4.2, we consider two types of delivery delay costs, which are linearly increasing and exponentially increasing. The linearly increasing cost type implies that the expected cost incurred due to the delayed delivery is proportional to the length of delayed time and for the exponentially increasing cost type, the expected cost is exponentially proportional to the delayed time.

4.1 Linearly increasing delivery delay cost ($c_p(t) = k_1 t$, $k_1 > 0$)

In this case, the following equations can be derived using equation (2):

$$\begin{aligned} y(t) &= d[EC(t)]/dt \\ &= c_1 + c_2\mu_Y ab \cdot \exp(-bt) \\ &\quad + c_3\mu_W ab[\exp(-bt_W) - 1] \cdot \exp(-bt) \\ &\quad + k_1 \cdot [1 - \exp(-t/\lambda_S)] \end{aligned}$$

and

$$u(t) = (k_1/\lambda_S) \cdot \exp[(b - 1/\lambda_S)t] \ .$$

It is noted that $u(t)$ is an increasing function of t when $b > 1/\lambda_S$ and a decreasing function of t when $b > 1/\lambda_S$ and $\lim_{t\to\infty} y(t) = c_1 + k_1$, which is positive. In addition, it is also possible to understand the graphical pattern of $y(t)$ by comparing the value of $u(0)$ with C. For example, if $b > 1/\lambda_S$ and $C \geq u(0)$, $y(t)$ is increasing after decreasing. Thus $y(t)$ can have four different types of graphical patterns: increasing, decreasing, increasing after decreasing, and decreasing after increasing type. For each type of $y(t)$, the method of selecting the optimal software release time, t^*, can be summarized as follows:

Theorem 1. Given c_1, c_2, c_3, μ_Y, μ_W, t_W, k_1, λ_s the optimal software release time, t^*,which minimizes the expected total software cost given in (1), can be determined as follows;

(a) Let $y(t)$ be an increasing or decreasing after increasing function. Then,

(1) if $y(0) \geq 0$ and $y(t) \geq 0$ for all t, then $t^* = 0$.

(2) if $y(0) < 0$ and $y(t) < 0$ for $t \in [0, t_1)$ and $y(t) \geq 0$ for $t \in [t_1, \infty)$, then

$$t^* = t_1 \text{ where } t_1 = y^{-1}(0).$$

(b) Let $y(t)$ be a decreasing function, then $t^* = 0$.

(c) Let $y(t)$ be an increasing after decreasing function. Then

(1) if $y(0) \geq 0$ and $y(t) > 0$ for all t, then $t^* = 0$.

(2) if $y(0) > 0$, $y(t) > 0$ for $t \in [0, t_2)$ and $y(t) \leq 0$ for $t \in [t_2, t_2')$, and $y(t) \geq 0$ for $t \in [t_2', \infty)$, then

$$t^* = 0 \text{ if } EC(0) \leq EC(t_2') \ ,$$
$$t^* = t_2' \text{ if } EC(0) > EC(t_2') \ ,$$

where $t_2' = y^{-1}(0)$.

(3) if $y(0) < 0$ and $y(t) < 0$ for $t \in [0, t_3)$ and $y(t) \geq 0$ for $t \in [t_3, \infty)$, then

$$t^* = t_3 \text{ where } t_3 = y^{-1}(0).$$

Proof.

We show the proof for case (c) only, where $y(t)$ is an increasing after decreasing function of t and $\lim_{t\to\infty} y(t) = c_1 + k_1 (>0)$.

(1) If $y(0) \geq 0$ and $y(t) > 0$ for all t, then $EC(t)$ is strictly increasing in t. Hence, $t^* = 0$ minimizes $EC(t)$.

(2) If $y(0) > 0$, there exist t_2, t_2' such that $y(t) > 0$ for any $t \in [0, t_2)$ and $y(t) \leq 0$ for any $t \in [t_2, t_2')$. Therefore, $t^* = 0$ if $EC(0) \leq EC(t_2')$ and $t^* = t_2'$ if $EC(0) > EC(t_2')$.

(3) If $y(0) < 0$, there exists a t_3 such that $y(t) < 0$ for any $t \in [0, t_3)$ and $y(t) \geq 0$ for any $t \in [t_3, \infty)$. Therefore, $t^* = t_3$, where $t_3 = y^{-1}(0)$.

The proofs of cases (a) and (b) are similar to that of (c) and so it is omitted.

4.2 Exponentially increasing delivery delay cost ($c_p(t) = k_2 \cdot [\exp(\gamma t - 1)]$, $\gamma > 0$, $k_2 > 0$)

Substituting $c_p(t) = k_2 \cdot [\exp(\gamma t - 1)]$ for $c_p(t)$ of the equation (2), we obtain

$$\begin{aligned} y(t) &= c_1 + c_2\mu_Y ab \cdot \exp(-bt) \\ &\quad + c_3\mu_W ab[\exp(-bt_W) - 1] \cdot \exp(-bt) \\ &\quad + k_2\gamma \cdot [\exp(\gamma t) - \exp(-t/\lambda_S)] / (1 + \lambda_S\gamma) \end{aligned}$$

and

$$\begin{aligned} u(t) &= k_2\gamma \cdot [1/\lambda_S \cdot \exp[(b - 1/\lambda_S)t] \\ &\quad + \gamma \exp[(b + \gamma)t]]/(1 + \lambda_S) \ . \end{aligned}$$

It is observed that for $b > 1/\lambda_S$, $u(t)$ is an increasing function of t and for $b < 1/\lambda_S$, $u(t)$ is an increasing function or an increasing after decreasing function of t. It is also noted that $\lim_{t\to\infty} y(t) = \infty$ and $\lim_{t\to\infty} u(t) = \infty$. Based on such observations, the graphic patterns of $y(t)$ can have one of three different monotonic properties: increasing, increasing after decreasing, and increasing initially, decreasing and then increasing. When $y(t)$ is increasing or increasing after decreasing function of t, the methods are the same as that described in cases (a) and (c) of Theorem 1. For the case of increasing initially, decreasing and then increasing, the optimal software release time, t^*, can be determined as in Theorem 2.

Theorem 2. Given c_1, c_2, c_3, μ_Y, μ_W, t_W, k_1, λ_s the optimal software release time, t^*, which minimizes the expected total software cost is determined as follows;

(a) Let $y(t)$ be an increasing initially, decreasing and then, increasing function of t. Then

(1) if $y(0) \geq 0$ and for all t $y(t) \geq 0$, then $t^* = 0$.

(2) if $y(0) > 0$, $y(t) > 0$ for $t \in [0, t_1)$, $y(t) \leq 0$ for $t \in [t_1, t_1')$, and $y(t) \geq 0$ for $t \in [t_1', \infty)$ then

$t^* = 0$ if $EC(0) \leq EC(t_1')$,

$t^* = t_1'$ if $EC(0) > EC(t_1')$,

where $t_1 = y^{-1}(0)$.

(3) if $y(0) < 0$ and $y(t) < 0$ for $t \in [0, t_2)$ and $y(t) \geq 0$ for $t \in [t_2, \infty)$, then

$t^* = t_2$ where $t_2 = y^{-1}(0)$.

(4) if $y(0) < 0$ and $y(t) < 0$ for $t \in [0, t_3)$, $y(t) \geq 0$ for $t \in [t_3, t_3')$, $y(t) \leq 0$ for $t \in [t_3', t_3'')$, $y(t) \geq 0$ for $t \in [t_3'', \infty)$, then

$t^* = t_3$ if $EC(t_3) \leq EC(t_3'')$,

$t^* = t_3''$ if $EC(t_3) > EC(t_3'')$.

Proof.

We prove the case (4) only. Since $y(t)$ is initially increasing, decreasing and then increasing, if $y(0) < 0$, then there exists t_3 , t_3' , t_3'' such that $y(t) < 0$ for any $t \in [0, t_3)$, $y(t) \geq 0$ for any $t \in [t_3, t_3')$, $y(t) \leq 0$ for any $t \in [t_3', t_3'')$ and $y(t) \geq 0$ for any $t \in [t_3'', \infty)$. Thus, it holds that $t^* = t_3$ if $EC(t_3) \leq EC(t_3'')$ and $t^* = t_3''$ if $EC(t_3) > EC(t_3'')$

The remaining cases can be proved similarly to that of (4) and thus the proofs are omitted.

5. Numerical examples

In this section, we present numerical examples to illustrate the proposed method in Section 4. For the purpose of numerical calculations, we apply our cost model to a set of real testing data, which is shown in <Table 1>. These data set is given in Misra (1983). Using the maximum likelihood estimate method, the parameters of Goel-Okumoto model can be easily estimated as $\hat{a} = 142.32$, $\hat{b} = 0.1246$.

<Table 1> Number of faults in 25 1-hour intervals and cumulative number of faults

Hour	1	2	3	4	5	6	7	8	9	10	11	12
no. of faults	27	16	11	10	11	7	2	5	3	1	4	7
cumulative no. of faults	27	43	54	64	75	82	84	89	92	93	97	104
Hour	14	15	16	17	18	19	20	21	22	23	24	25
no. of faults	5	5	6	0	5	1	1	2	1	2	1	1
cumulative no. of faults	111	116	122	122	127	128	129	131	132	134	135	136

As for the coefficients specified in the cost model considered, we assume the following parameter values:

$c_1 = 1$, $c_2 = 5$, $c_3 = 10$, $t_W = 10$, $\mu_Y = 0.1$,

$\mu_W = 0.3$, $\lambda_s = 10$.

<Tables 2 and 3> show the numerical results with regard to the optimal software release time and the corresponding quantities for two types of $c_p(t)$ discussed in Section 4.

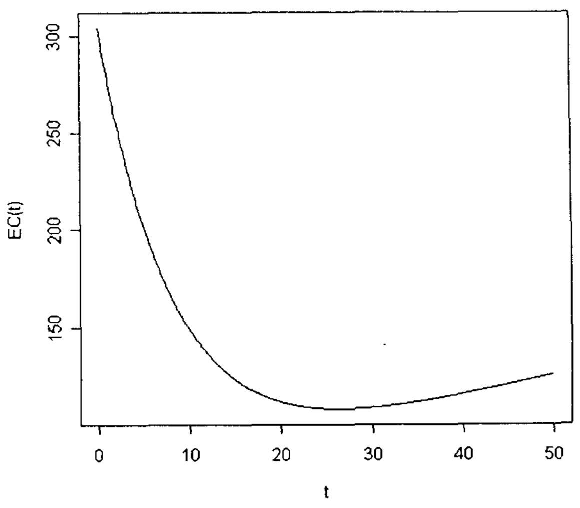
<Table 2> Optimal software release time when $c_p(t) = k_1 t$.

k_1	Optimal release time t^*	Expected total software cost $EC(t^*)$	Expected number of detected faults $m(t^*)$	fault detection rate $m(t^*)/a$
0.1	26.3208	107.955	136.962	0.962356
0.5	24.0248	114.325	135.188	0.949888
1	21.9308	121.293	133.062	0.934949
2	19.0542	132.987	129.071	0.906907
3	17.0796	142.655	125.375	0.880939
4	15.5943	150.92	121.93	0.856734
5	14.4152	158.141	118.704	0.834061

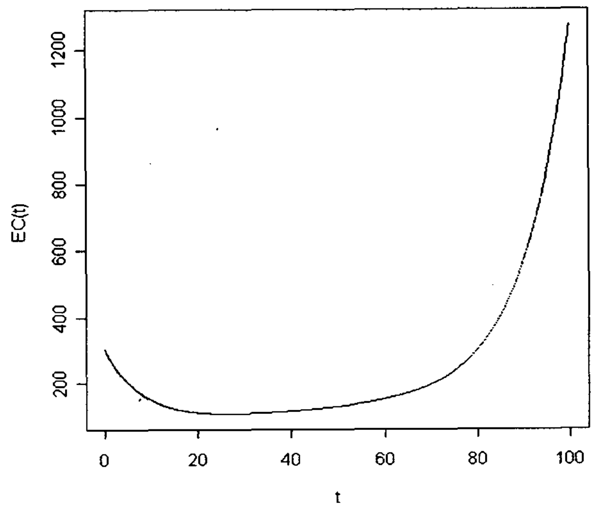
<Table 3> Optimal software release time when $c_p(t) = k_2[\exp(0.1t-1)]$, with $\gamma = 0.1$.

k_2	Optimal release time t^*	Expected total software cost $EC(t^*)$	Expected number of detected faults $m(t^*)$	fault detection rate $m(t^*)/a$
0.1	26.4901	106.852	137.074	0.963141
0.5	24.9366	109.086	135.954	0.95527
1	23.6561	111.459	134.853	0.947532
2	21.9751	115.432	133.113	0.935307
3	20.7278	118.79	131.698	0.925365
4	19.9523	121.757	130.474	0.916762
5	19.2431	124.444	129.379	0.909072

From Tables 2 and 3, we observe that as the parameter values k_1 and k_2 in $c_p(t)$ increase, then the expected total cost $EC(t)$ increases. However, the optimal software release time t^* and the expected number of detected software faults $m(t^*)$ decreases for both cases. <Figures 1 and 2> graph the expected total software cost as a function of t for two types of delayed delivery costs.



<Figure 1> Expected total software cost, $EC(t)$ when $c_p(t) = 0.1t$



<Figure 2> Expected total software cost, $EC(t)$ when $c_p(t) = 1/10 \cdot [\exp(0.1t-1)]$

References

[1] A.L. Goel and K. Okumoto (1979), Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Transactions on Reliability*, vol. 28, 206-211.

[2] C.H. Lee, K.H. Nam and D.H. Park (2002), Software profit model under imperfect debugging and optimal software release policy, *IEICE Transactions on Information and Systems*, vol. E85-D No.5, 833-838.

[3] P.N. Misra (1983), Software reliability

analysis, *IBM Systems Journal*, vol. 22, pp. 262-270.
[4] H. Pham (1996), A software cost model with imperfect debugging, random life cycle and penalty cost, *International Journal of Systems Science*, vol. 27, 455-463.
[5] S. Yamada, K. Tokuno and S. Osaki (1992), Imperfect debugging models with fault introduction rate for software reliability assessment, *International Journal of Systems Science*, vol. 23, 2241-2252.
[6] S. Yamada and S. Osaki (1985), Cost-reliability optimum release policies for software systems, *IEEE Transactions on Reliability*, vol. 34, 422-424.
[7] W.Y. Yun and D.S. Bai (1990), Optimum software release policy with random life cycle, *IEEE Transactions on Reliability*, vol.39, 167-170.