The Korean Journal of Applied Statistics (2010)
23(3), 573–584

The Software Reliability Growth Models for Software Life-Cycle Based on NHPP

Kyung H. Nam[1] · Do Hoon Kim[2]

[1]Department of Applied Information Statistics, Kyonggi University
[2]Department of Applied Information Statistics, Kyonggi University

(Received July 2009; accepted May 2010)

Abstract

This paper considers the differences in the software execution environments in the testing phase and the operational phase to determine the optimal release time and warranty period of software systems. We formulate equations for the total expected software cost until the end of the software life cycle based on the NHPP. In addition, we derive the optimal release time that minimizes the total expected software cost for an imperfect debugging software reliability model. Finally, we analyze the sensitivity of the optimal testing and maintenance design related to variation of the cost model parameters based on the fault data observed in the actual testing process, and discuss the quantitative properties of the proposed model.

Keywords: Software reliability growth model, nonhomogeneous Poisson process, software release time, warranty period, software maintenance activity.

1. Introduction

Every software project manager is expected to release fault-free computer software to his/her customers. Due to the complicity of current software, it has been increasingly difficult for development managers to produce highly reliable and effective software systems. In the context of the software development, penalty costs for software failures are even more significant. As a result, the determination of the software release time is an important factor in the success or failure of software. Okumoto and Goel (1980) assumed that the number of software faults detected in the software testing phase conforms to an exponential software reliability model (Goel and Okumoto, 1979) based on the NHPP and derived the optimal release time minimizing the total expected cost. Koch and Kubat (1983) assumed the Jelinski and Moranda (1972) model for the software fault detection process and discussed a similar problem.

It is not easy to prevent the occurrence of failures caused by software faults after release since detecting and removing all of the faults latent in the software in the actual testing process are extremely difficult. In many cases, the development manager must investigate the causes of software

This work was supported by Kyonggi University Research Grant 2007.
[1]Corresponding author: Professor, Department of Applied Information Statistics, Kyonggi University, Suwon 443-760, Korea. E-mail: knam@kyonggi.ac.kr

failures that occurred after the software release based on the maintenance and warranty contract with the user to detect and remove the faults. To perform maintenance in the operational phase (after the release), the software development manager is requested to reduce the management costs in the operational phase and effectively utilize the human resources in contrast that require continued support by the development project team. Although the length of period for continued maintenance by the development project team has the same importance as the release time, but they have been considered in the little literature.

Yamada (1994) and Kimura *et al.* (1999) considered the problem of determining the optimal release time while assuming the software warranty period to be a random variable. Pham and Zhang (1999) proposed a software cost model that simultaneously considered the warranty and the risk after release. However, the operational maintenance of the software is provided as a service after release by the development manager and must be designed by the maintenance contract itself and the product delivery date. Dohi *et al.* (2000) and Rinsaka and Sandoh (1999) determine the optimal warranty period that minimizes the total expected software cost based on the assumption that the debugging process in the testing process is described by an NHPP. However, most of the research does not fully consider the differences in the debugging environments in the testing phase and the operational phase, and present their discussions based on a simple cost structure.

This paper proposes a model for the differences in the software execution environments in the testing phase and the operational phase. We consider that the actual maintenance activities are ineffective during the warranty period. We formulate the mathematical equations for the total expected software cost until the end of the software life cycle based on the NHPP. In addition, we derive the optimal release time and warranty period that minimizes the total expected software cost for an imperfect debugging software reliability model. Finally, we analyze the sensitivity of the optimal testing and maintenance design related to the variation of the cost model parameters based on the fault data observed in the actual testing process, and discuss the quantitative properties of the proposed model.

2. Model Descriptions

2.1. Assumptions

We consider two periodic software maintenance policies in association with an imperfect debugging software reliability growth model. We construct the several costs required in the software development process for failure-occurrence time data to formulate the total expected software cost. The failure-occurrence time is the time-interval between software failure-occurrences. Throughout this paper, we postulate the following assumptions.

Assumption 2.1. *The software system starts to test at time* $t = 0$.

Assumption 2.2. *Software development manager conducts the maintenance activities(e.g. patch, update) during the warranty period* t_w *after the software release.*

Assumption 2.3. *The periodic software maintenance is done at periodic time* kt_w/N, $(k = 1, 2, \ldots, N$, $t_w \geq 0)$, *and is stopped the maintenance activities at the* N^{th} *periodic software maintenance.*

Assumption 2.4. *The life cycle* $t_L > 0$ *of a software product is known.*

Assumption 2.5. *A new fault may be introduced into the software system due to an imperfect debugging.*

Let $\{N(t),\ t \geq 0\}$ be a counting process representing the cumulative number of software faults detected or removed up to time t. The detection time of each software fault can be formulated as an NHPP as follows:

$$\Pr\{N(t) = n\} = \frac{\{m(t)\}^n}{n!} \exp\{-m(t)\}, \quad (n = 0, 1, 2, \ldots), \tag{2.1}$$

where $m(t)$ is the mean value function of the NHPP, that is, the expected cumulative number of faults detected up to time t.

Pham (1993) introduced an NHPP SRGM that is subject to an imperfect debugging. He assumed that the detected faults are removed and then there is a possibility to introduce new faults with constant rate β. Let $a(t)$ be the number of faults to be eventually detected, where $a(t)$ is the time-dependent fault content function. The mean value function $m(t)$ can be given as the solution of the following system of differential equations:

$$\begin{aligned} &\frac{\partial m(t)}{\partial t} = b[a(t) - m(t)], \qquad \frac{\partial a(t)}{\partial t} = \beta \frac{\partial m(t)}{\partial t}, \\ &a(0) = a,\ m(0) = 0, \end{aligned} \tag{2.2}$$

where a is the number of faults to be eventually detected. Solving the Equation (2.2), we obtain the mean value function $m(t)$ and instantaneous fault detection rate $\lambda(t)$ as follows:

$$\begin{aligned} m(t) &= \frac{a}{1-\beta}\left[1 - e^{-(1-\beta)bt}\right], \\ \lambda(t) &= abe^{-(1-\beta)bt}. \end{aligned} \tag{2.3}$$

To determine the optimal release time and warranty period of software systems, we propose two periodic software maintenance policies in terms of the behavior of instantaneous fault detection rate with an imperfect debugging SRGM.

Case 1. Let us assume that after the software release, the software reliability growth occurs(assuming that we correct only major faults that will improve the reliability of the software). That is, the instantaneous fault detection rate after the release time assumes the decreasing function.

Case 2. Let us assume that during the warranty period, the software reliability growth does not occur(assuming that we correct only minor faults). That is, the instantaneous fault detection rate after the release follows that of the release time point, is continued as constant.

In Figure 2.1, t_0 denotes the software release time, t_w denotes the software warranty period, t_L denotes the software life cycle and N denotes the number of maintenance activities during the warranty period and $0 < t_1(= t_0 + t_w/N) < t_2(= t_0 + 2t_w/N) < \cdots < t_N(= t_0 + t_w)$.

The instantaneous fault detection rates of the proposed periodic maintenance policy are as follows.

1. Policy 1 based on Case(1)

$$\lambda_{pm}(t) = \lambda(t), \quad 0 < t \leq t_0 + t_L. \tag{2.4}$$

2. Policy 2 based on Case(2)

$$\lambda_{pm}(t) = \begin{cases} \lambda(t), & 0 < t \leq t_0, \\ \lambda(t_0), & t_0 < t \leq t_0 + t_L. \end{cases} \tag{2.5}$$

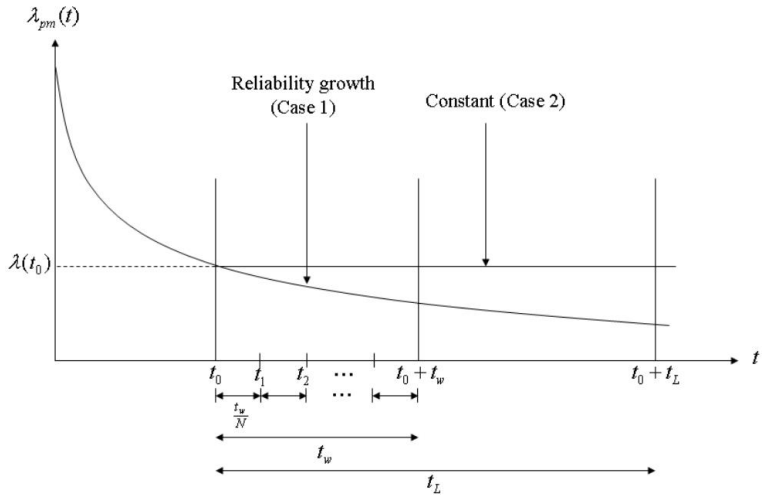

Figure 2.1. Software reliability growth during the warranty period of ineffective maintenance policies

2.2. Total expected software cost

To formulate the total expected software cost based on data related to the number of faults detected in the software development process, the cost parameters are defined as follows.

$c_0(>0)$: cost to remove each fault in the testing phase.

$c_w(>0)$: cost to remove each fault during the warranty period.

$c_L(>0)$: cost to remove each fault after the warranty period.

$c_p(>0)$: unit maintenance cost.

$c_t(>0)$: testing cost per unit time.

Based on the Pham (1993)'s NHPP model described in Section 2.1, we formulate the total expected software cost in the software development project.

Several expected costs that are required during the software development project can be determined as follows:

1) The expected cumulative number of software faults removed in the testing phase $(0, t_0)$ is given by

$$E\left[N(t_0)\right] = m(t_0), \tag{2.6}$$

where $N(t_0)$ is the number of faults detected up to time t_0. Hence, the expected cost to remove all faults detected by time t_0, $\mathrm{EC}_0(t_0,\, t_w)$, can be expressed as

$$\mathrm{EC}_0(t_0, t_w) = c_0 \cdot E\left[N(t_0)\right] = c_0 m(t_0). \tag{2.7}$$

2) The expected cumulative number of software faults removed during warranty period $[t_0, t_0 + t_w)$ is given by

$$E\left[N(t_0 + t_w) - N(t_0)\right] = \left[m(t_0 + t_w) - m(t_0)\right]. \tag{2.8}$$

Hence, the expected cost to remove all faults detected by time $[t_0,\, t_0 + t_w)$, $\mathrm{EC}_w(t_0,\, t_w)$, can be expressed as

$$\mathrm{EC}_w(t_0, t_w) = c_w \cdot E\left[N(t_0 + t_w) - N(t_0)\right] = c_w\left[m(t_0 + t_w) - m(t_0)\right]. \tag{2.9}$$

3) The expected cumulative number of software faults removed after warranty period $[t_0 + t_w,\, t_0 + t_L)$ is given by

$$E\left[N(t_0 + t_L) - N(t_0 + t_w)\right] = \left[m(t_0 + t_L) - m(t_0 + t_w)\right]. \tag{2.10}$$

Hence, the expected cost to remove all faults detected by time $[t_0 + t_w,\, t_0 + t_L)$, $\mathrm{EC}_L(t_0, t_w)$, can be expressed as

$$\mathrm{EC}_L(t_0, t_w) = c_L \cdot E\left[N(t_0 + t_L) - N(t_0 + t_w)\right] = c_L\left[m(t_0 + t_L) - m(t_0 + t_w)\right]. \tag{2.11}$$

4) Cost to do patch, $\mathrm{EC}_p(t_0, t_w)$ is proportional to the number of maintenance activities

$$\mathrm{EC}_p(t_0, t_w) = c_p \cdot N. \tag{2.12}$$

5) Testing cost, $\mathrm{EC}_t(t_0, t_w)$ is a linear function of time t_0 and t_w

$$\mathrm{EC}_t(t_0, t_w) = c_t(t_0 + t_w). \tag{2.13}$$

Therefore, the total expected software cost, $\mathrm{EC}(t_0, t_w)$ can be expressed as

$$\begin{aligned}\mathrm{EC}(t_0, t_w) = {} & c_0 m(t_0) + c_w\left[m(t_0 + t_w) - m(t_0)\right] \\ & + c_L\left[m(t_0 + t_L) - m(t_0 + t_w)\right] + c_p N + c_t(t_0 + t_w).\end{aligned} \tag{2.14}$$

By the total expected software cost of proposed maintenance policies in Case 1 and 2 replacing $\lambda(t)$ with $abe^{-(1-\beta)bt}$ in Equation (2.3), we can be calculated as follows.

1. Policy 1 based on Case(1)

$$\begin{aligned}\mathrm{EC}_1(t_0, t_w) = {} & c_0 \int_0^{t_0} abe^{-(1-\beta)bt}dt + c_w \int_{t_0}^{t_0+t_w} abe^{-(1-\beta)bt}dt + c_L \int_{t_0+t_w}^{t_0+t_L} abe^{-(1-\beta)bt}dt \\ & + c_p N + c_t(t_0 + t_w).\end{aligned} \tag{2.15}$$

2. Policy 2 based on Case(2)

$$\begin{aligned}\mathrm{EC}_2(t_0, t_w) = {} & c_0 \int_0^{t_0} abe^{-(1-\beta)bt}dt + c_w \int_{t_0}^{t_0+t_w} abe^{-(1-\beta)bt_0}dt + c_L \int_{t_0+t_w}^{t_0+t_L} abe^{-(1-\beta)bt_0}dt \\ & + c_p N + c_t(t_0 + t_w).\end{aligned} \tag{2.16}$$

In the next chapter, we discuss the optimal release time t_0^* and the optimal warranty period t_w^* minimizing the total expected software cost.

3. Optimal Software Release Time and Warranty Period

We now derive the optimal release time and the optimal warranty period for proposed policies that minimize the total expected software cost, $\mathrm{EC}_1(t_0, t_w)$, $\mathrm{EC}_2(t_0, t_w)$, respectively.

3.1. Policy 1 based on Case(1)

To show the existence and uniqueness of the optimal release time t_0^* and the optimal warranty period t_w^*, which minimize $\mathrm{EC}_1(t_0, t_w)$, we rewrite the total expected software cost of Equation (2.15) as follows.

$$\begin{aligned}\mathrm{EC}_1(t_0, t_w) = &\frac{ac_0}{1-\beta}\left[1 - e^{-(1-\beta)bt_0}\right] + \frac{ac_w}{1-\beta}\left[e^{-(1-\beta)bt_0}\left(1 - e^{-(1-\beta)bt_w}\right)\right] \\ &+ \frac{ac_L}{1-\beta}\left[e^{-(1-\beta)bt_0}\left(e^{-(1-\beta)bt_w} - e^{-(1-\beta)bt_L}\right)\right] + c_pN + c_t(t_0 + t_w). \end{aligned} \tag{3.1}$$

The following assumptions are set:

(A-I) $c_L > c_w > c_0$

(A-II) $c_w\left(1 - e^{-(1-\beta)bt_L}\right) > c_0$

(A-III) $c_w\left(1 - e^{-(1-\beta)bt_w}\right) + c_L\left(e^{-(1-\beta)bt_w} - e^{-(1-\beta)bt_L}\right) > c_0$

and let

$$Q_1(t_w) = ab\left[c_0 - c_w\left(1 - e^{-(1-\beta)bt_w}\right) - c_L\left(e^{-(1-\beta)bt_w} - e^{-(1-\beta)bt_L}\right)\right] + c_0. \tag{3.2}$$

Theorem 3.1. *Given $c_0, c_w, c_L, c_p, c_t, N, t_w, t_L$, the optimal release time t_0^*, which minimizes the total expected software cost, $EC_1(t_0, t_w)$ can be determined as follows based on A-I to A-III.*

Case 1. When $Q_1(t_w) < 0$, a finite and unique $t_0^*(> 0)$ that minimizes the total expected software cost $EC_1(t_0, t_w)$ exists:

$$\begin{aligned}EC_1(t_0^*, t_w) = &\frac{ac_0}{1-\beta}\left[1 - e^{-(1-\beta)bt_0^*}\right] + \frac{ac_w}{1-\beta}\left[e^{-(1-\beta)bt_0^*}\left(1 - e^{-(1-\beta)bt_w}\right)\right] \\ &+ \frac{ac_L}{1-\beta}\left[e^{-(1-\beta)bt_0^*}\left(e^{-(1-\beta)bt_w} - e^{-(1-\beta)bt_L}\right)\right] + c_pN + c_t\left(t_0^* + t_w\right). \end{aligned} \tag{3.3}$$

Case 2. When $Q_1(t_w) > 0$, $t_0^* = 0$ and

$$EC_1(t_0^*, t_w) = \frac{ac_w}{1-\beta}\left[1 - e^{-(1-\beta)bt_w}\right] + \frac{ac_L}{1-\beta}\left[e^{-(1-\beta)bt_w} - e^{-(1-\beta)bt_L}\right] + c_pN + c_tt_w. \tag{3.4}$$

The software debugging becomes optimal by only an acceptance test by the user in the operational phase.

Proof.

$$\begin{aligned}\frac{\partial \mathrm{EC}_1(t_0, t_w)}{\partial t_0} = &abc_0e^{-(1-\beta)bt_0} - abc_we^{-(1-\beta)bt_0}\left(1 - e^{-(1-\beta)bt_w}\right) \\ &- abc_Le^{-(1-\beta)bt_0}\left(e^{-(1-\beta)bt_w} - e^{-(1-\beta)bt_L}\right) + c_t. \end{aligned} \tag{3.5}$$

Let $\delta_1(t_0, t_w)$ denote the right-hand side of Equation (3.5), then

$$\delta_1(0, t_w) = ab\left[c_0 - c_w\left(1 - e^{-(1-\beta)bt_w}\right) - c_L\left(e^{-(1-\beta)bt_w} - e^{-(1-\beta)bt_L}\right)\right] + c_t \tag{3.6}$$

and

$$\lim_{t_0 \to +\infty} \delta_1(t_0, t_w) = c_t. \tag{3.7}$$

Differentiating Equation (3.5) with respect to t_0, we obtain

$$\frac{\partial^2 \mathrm{EC}_1(t_0,t_w)}{\partial^2 t_0} = ab^2(1-\beta)e^{-(1-\beta)bt_0}\left[-c_0+c_w\left(1-e^{-(1-\beta)bt_w}\right)+c_L\left(e^{-(1-\beta)bt_w}-e^{-(1-\beta)bt_L}\right)\right]. \quad (3.8)$$

Letting $\gamma_1(t_0,t_w)$ denote the right-hand side of Equation (3.8),

$$\gamma_1(0,t_w) = ab^2(1-\beta)\left[-c_0+c_w\left(1-e^{-(1-\beta)bt_w}\right)+c_L\left(e^{-(1-\beta)bt_w}-e^{-(1-\beta)bt_L}\right)\right]. \quad (3.9)$$

Letting $A_1(t_w)$ denote the right-hand side of Equation (3.9), $A_1(0) > 0$ is equivalent to $c_L(1 - e^{-(1-\beta)bt_L}) > c_0$, $A_1(t_L) > 0$ is equivalent to $c_w(1 - e^{-(1-\beta)bt_L}) > c_0$. Therefore, from assumption A-I, $A_1(0) > A_1(t_L)$ holds. Furthermore, $A_1'(t_w)$ becomes

$$A_1'(t_w) = -(c_L - c_w)ab^3(1-\beta)^2 e^{-(1-\beta)bt_w} < 0, \quad (3.10)$$

$\gamma_1(0,t_w) > 0$. Next,

$$\lim_{t_0\to+\infty}\gamma_1(t_0,t_w) = 0 \quad (3.11)$$

$\partial\gamma_1(t_0,t_w)/\partial t_0 < 0$ is equivalent to

$$c_w\left(1-e^{-(1-\beta)bt_w}\right)+c_L\left(e^{-(1-\beta)bt_w}-e^{(1-\beta)bt_L}\right) > c_0. \quad (3.12)$$

□

Theorem 3.2. *Given $c_0, c_w, c_L, c_p, c_t, N, t_0, t_L$, the optimal warranty period t_w^*, which minimizes the total expected software cost $EC_1(t_0,t_w)$ based on A-I is determined by the following.*

Case 1. When $c_t \geq abe^{-(1-\beta)bt_0}(c_L - c_w)$, $t_w^* = 0$ minimizes $EC_1(t_0,t_w)$.

$$EC_1(t_0,t_w^*) = \frac{ac_0}{1-\beta}\left[1-e^{-(1-\beta)bt_0}\right]+\frac{ac_L}{1-\beta}\left[e^{-(1-\beta)bt_0}\left(1-e^{-(1-\beta)bt_L}\right)\right]+c_pN+c_tt_0. \quad (3.13)$$

Case 2. When $c_t < abe^{-(1-\beta)bt_0}(c_L - c_w)$ and $c_t > abe^{-(1-\beta)b(t_0+t_L)}(c_L - c_w)$, a finite and unique $t_w^*(0 < t_w < t_L)$ that minimizes the total expected software cost $EC_1(t_0,t_w)$ exists.

$$\begin{aligned} EC_1(t_0,t_w^*) = {} & \frac{ac_0}{1-\beta}\left[1-e^{-(1-\beta)bt_0}\right]+\frac{ac_w}{1-\beta}\left[e^{-(1-\beta)bt_0}\left(1-e^{-(1-\beta)bt_w^*}\right)\right] \\ & +\frac{ac_L}{1-\beta}\left[e^{-(1-\beta)bt_0}\left(e^{-(1-\beta)bt_w^*}-e^{-(1-\beta)bt_L}\right)\right]+c_pN+c_t(t_0+t_w^*). \end{aligned} \quad (3.14)$$

Case 3. When $c_t \leq abe^{-(1-\beta)b(t_0+t_L)}(c_L - c_w)$, $t_w^* = t_L$ minimizes $EC_1(t_0,t_w)$.

$$\begin{aligned} EC_1(t_0,t_w^*) = {} & \frac{ac_0}{1-\beta}\left[1-e^{-(1-\beta)bt_0}\right]+\frac{ac_w}{1-\beta}\left[e^{-(1-\beta)bt_0}\left(1-e^{-(1-\beta)bt_L}\right)\right] \\ & +c_pN+c_t(t_0+t_L). \end{aligned} \quad (3.15)$$

Proof.

$$\frac{\partial \mathrm{EC}_1(t_0,t_w)}{\partial t_w} = abe^{-(1-\beta)b(t_0+t_w)}(c_w - c_L)+c_t. \quad (3.16)$$

Let $\xi_1(t_0,t_w)$ denote the right-hand side of Equation (3.16). Then

$$\xi_1(t_0,0) = abe^{-(1-\beta)bt_0}(c_w - c_L)+c_t \quad (3.17)$$

and

$$\lim_{t_w \to +\infty} \xi_1(t_0, t_w) = c_t. \tag{3.18}$$

Differentiating Equation (3.17) with respect to t_w, we obtain

$$\frac{\partial \xi_1(t_0, t_w)}{\partial t_w} = ab^2(1-\beta)e^{-(1-\beta)b(t_0+t_w)}(c_L - c_w). \tag{3.19}$$

Let $\eta_1(t_0, t_w)$ denote the right-hand side of Equation (3.16). According to the assumption A-I, $\eta_1(t_0, t_w) > 0$. □

Now, we introduce the algorithm presented by Rinsaka and Dohi (2006) for simultaneously determining the optimal release time and the optimal warranty period which minimizes $\mathrm{EC}_1(t_0, t_w)$ as discussed in maintenance policy of hardware product (Mi, 1994; Cha, 2000, 2001).

Algorithm

Step 1. If (t_0, t_w) which satisfied (3.20) exists for $t_0 \geq 0$ and $0 \leq t_w \leq t_L$, go to Step 2. If not, execute Step 3.

$$\frac{\partial \mathrm{EC}_1(t_0, t_w)}{\partial t_0} = \frac{\partial \mathrm{EC}_1(t_0, t_w)}{\partial t_w} = 0. \tag{3.20}$$

Step 2. The Hessian is defined as

$$H(t_0, t_w) = \frac{\partial^2 \mathrm{EC}_1(t_0, t_w)}{\partial t_0^2} \frac{\partial^2 \mathrm{EC}_1(t_0, t_w)}{\partial t_w^2} - \left(\frac{\partial^2 EC_1(t_0, t_w)}{\partial t_0 \partial t_w} \right)^2. \tag{3.21}$$

If $H(t_0, t_w) > 0$ and $\partial^2 \mathrm{EC}_1(t_0, t_w)/\partial t_0^2 > 0$, the solution found in Step 1 becomes the joint optimal policy (t_0^{**}, t_w^{**}), and the algorithm ends. If not, execute Step 3.

Step 3. Determine $t_w^*(0 \leq t_w^* \leq t_L)$ that satisfies $\partial \mathrm{EC}_1(0, t_w)/\partial t_w = 0$, and set

$$\mathrm{EC}_1^{(1)}(t_0^*, t_w^*) = \mathrm{EC}_1(0, t_w^*).$$

Step 4. Determine $t_0^*(0 \leq t_0^* < \infty)$ that satisfies $\partial \mathrm{EC}_1(t_0, 0)/\partial t_0 = 0$, and set

$$\mathrm{EC}_1^{(2)}(t_0^*, t_w^*) = \mathrm{EC}_1(t_0^*, 0).$$

Step 5. Determine $t_0^*(0 \leq t_0^* < \infty)$ that satisfies $\partial \mathrm{EC}_1(t_0, t_L)/\partial t_0 = 0$, and set

$$\mathrm{EC}_1^{(3)}(t_0^*, t_w^*) = \mathrm{EC}_1(t_0^*, t_L^*).$$

Step 6. The minimum total expected software cost is obtained from $\mathrm{EC}_1(t_0^{**}, t_w^{**}) = \min_{i=1,2,3} \mathrm{EC}_1^{(i)}(t_0^*, t_w^*)$ and the corresponding (t_0^{**}, t_w^{**}) becomes the joint optimal policy.

3.2. Policy 2 based on Case(2)

By utilizing the similar technique as in Policy 1, we find the optimal release time t_0^* and the optimal warranty period t_w^*, which minimize $\mathrm{EC}_2(t_0, t_w)$. Rewriting the total expected software cost of

Equation (2.16), we have

$$\mathrm{EC}_2(t_0,t_w)=\frac{ac_0}{1-\beta}\left[1-e^{-(1-\beta)bt_0}\right]+abe^{-(1-\beta)bt_0}[c_wt_w+c_Lt_L-c_Lt_w]+c_pN+c_t(t_0+t_w). \quad (3.22)$$

The following assumptions are set:

(B-I) $c_L > c_w > c_0$

(B-II) $b(1-\beta)c_wt_L > c_0$

(B-III) $b(1-\beta)\left[c_wt_w + c_Lt_L - c_Lt_w\right] > c_0$

and let

$$Q_2(t_w) = ab\left[c_0 - b(1-\beta)\left(c_wt_w + c_Lt_L - c_Lt_w\right)\right] + c_t. \quad (3.23)$$

Theorem 3.3. *Given $c_0, c_w, c_L, c_p, c_t, N, t_w, t_L$, the optimal release time t_0^*, which minimizes the total expected software cost $EC_2(t_0,t_w)$ can be determined as follows based on B-I to B-III.*

Case 1. When $Q_2(t_w) < 0$, a finite and unique $t_0^*(>0)$ that minimizes the total expected software cost $EC_2(t_0,t_w)$ exists:

$$EC_2(t_0^*,t_w) = \frac{ac_0}{1-\beta}\left[1-e^{-(1-\beta)bt_0^*}\right] + abe^{-(1-\beta)bt_0^*}\left[c_wt_w + c_Lt_L - c_Lt_w\right] + c_pN + c_t(t_0^*+t_w). \quad (3.24)$$

Case 2. When $Q_2(t_w) \geq 0$, $t_0^* = 0$ and

$$EC_2(t_0^*,t_w) = ab[c_wt_w + c_Lt_L - c_Lt_w] + c_pN + c_tt_w. \quad (3.25)$$

The software debugging becomes optimal by only an acceptance test by the user in the operational phase.

Proof. Omitted. □

Theorem 3.4. *Given $c_0, c_w, c_L, c_p, c_t, N, t_0, t_L$, the optimal warranty period t_w^*, which minimizes the total expected software cost $EC_2(t_0,t_w)$ is determined by the following.*

Case 1. When $c_t \geq abe^{-(1-\beta)bt_0}(c_L - c_w)$, $t_w^* = 0$ minimizes $EC_2(t_0,t_w)$.

$$EC_2(t_0,t_w^*) = \frac{ac_0}{1-\beta}\left[1-e^{-(1-\beta)bt_0}\right] + abe^{-(1-\beta)bt_0}c_Lt_L + c_pN + c_tt_0. \quad (3.26)$$

Case 2. When $c_t \leq abe^{-(1-\beta)bt_0}(c_L - c_w)$, $t_w^* = t_L$ minimizes $EC_2(t_0,t_w)$.

$$EC_2(t_0,t_w^*) = \frac{ac_0}{1-\beta}\left[1-e^{-(1-\beta)bt_0}\right] + abe^{-(1-\beta)bt_0}c_wt_L + c_pN + c_t(t_0+t_L). \quad (3.27)$$

Proof. Omitted. □

The algorithm presented by Rinsaka and Dohi (2006) for simultaneously determining the optimal release time and the optimal warranty period which minimize $\mathrm{EC}_2(t_0,t_w)$ is similar to previous algorithm.

Table 4.1. Software failure occurrence time data from System T1

software failure times (CPUs)							
3	1846	5324	10258	15806	26770	42296	56485
33	1872	5389	10491	16185	27753	42296	56560
146	1986	5565	10625	16229	28460	45406	57042
227	2311	5623	10982	16358	28493	46653	62551
342	2366	6080	11175	17168	29361	47596	62651
351	2608	6380	11411	17458	30085	48296	62661
353	2676	6477	11442	17758	32408	49171	63732
444	3098	6740	11811	18287	35338	49416	64106
556	3278	7192	12559	18568	36799	50145	64893
571	3288	7447	12559	18728	37642	52042	71043
709	4434	7644	12791	19556	37654	52489	74364
759	5036	7837	13121	20567	37915	52875	75409
836	5049	7843	13486	21012	39715	53321	76057
860	5085	7922	14708	21308	40580	53443	81542
968	5089	8738	15251	23063	42015	54433	82702
1056	5089	10089	15261	24127	42045	55381	84566
1726	5097	10237	15277	25910	42188	56463	88682

Table 4.2. Optimal release time and its corresponding cost for Policy 1

c_t	$t_w = 200$		$t_w = 210$		$t_w = 220$		$t_w = 230$		$t_w = 250$	
	t_0^*	$\mathrm{EC}_1(t_0^*, t_w)$	t_0^*	$\mathrm{EC}_1(t_0^*, t_w)$	t_0^*	$\mathrm{EC}_1(t_0^*, t_w)$	t_0^*	$\mathrm{EC}_1(t_0^*, t_w)$	t_0^*	$\mathrm{EC}_1(t_0^*, t_w)$
0.20	982.29	1815.19	923.48	1803.87	863.58	1790.37	763.39	1772.52	628.26	1745.74
0.21	752.12	1823.45	709.19	1814.79	624.31	1805.21	531.31	1785.57	389.51	1755.21
0.22	562.82	1833.28	481.53	1822.82	423.10	1813.28	333.25	1794.82	153.12	1758.74
0.23	354.14	1842.92	298.19	1830.67	216.54	1820.12	120.51	1812.36	0	1760.63
0.24	152.21	1849.11	98.28	1836.52	23.56	1826.65	0	1809.54	0	1762.86
0.25	0	1953.65	0	1840.88	0	1829.43	0	1811.17	0	1764.64

4. Numerical Example

In this chapter, we apply proposed cost models to a set of real testing data of System T1 (Musa, 1979). This system T1 data is summarized as a software failure occurrence time data per fault. Using the maximum likelihood estimate method, the estimates of parameter of Pham (1993) model can be obtained:

$$\hat{a} = 132.6, \qquad \hat{b} = 3.52 \times 10^{-5}, \qquad \hat{\beta} = 0.1337.$$

The estimated mean value function becomes:

$$\hat{m}(t) = 156.0648 \left[1 - e^{-0.00003049t}\right].$$

We assume the following parameters to obtain optimal release time and warranty period in the proposed cost model as

$$c_0 = 1.0, \qquad c_w = 3.0, \qquad c_L = 20.0, \qquad c_p = 2.0, \qquad t_L = 1000 \quad \text{and} \quad N = 60.$$

The optimal release times are then calculated for different values of testing cost per unit time c_t and warranty period t_w. The results are tabulated and shown in Tables 4.2 and 4.3, respectively.

From Tables 4.2 and 4.3, it can be seen that when the testing cost per unit time c_t for fixed the warranty period t_w increases, the optimal release time t_0^* is shorten and the total expected software

Table 4.3. Optimal release time and its corresponding cost for Policy 2

c_t	$t_w = 200$		$t_w = 210$		$t_w = 220$		$t_w = 230$		$t_w = 250$	
	t_0^*	$\mathrm{EC}_2(t_0^*, t_w)$	t_0^*	$\mathrm{EC}_2(t_0^*, t_w)$	t_0^*	$\mathrm{EC}_2(t_0^*, t_w)$	t_0^*	$\mathrm{EC}_2(t_0^*, t_w)$	t_0^*	$\mathrm{EC}_2(t_0^*, t_w)$
0.20	1034.35	1798.78	959.09	1785.72	882.76	1772.45	805.28	1758.96	646.86	1731.28
0.21	809.42	1809.99	734.15	1796.28	657.80	1782.35	580.33	1768.18	421.91	1739.11
0.22	594.94	1819.00	519.67	1804.64	443.32	1790.05	365.86	1775.20	207.43	1744.75
0.23	389.99	1825.92	314.73	1810.91	238.38	1795.65	160.91	1780.13	2.49	1748.29
0.24	193.77	1830.83	118.51	1815.17	42.16	1799.24	0	1783.08	0	1750.92
0.25	5.56	1833.82	0	1817.63	0	1801.49	0	1785.38	0	1753.29

Table 4.4. Optimal warranty period and its corresponding for Policy 1

c_t	$t_0 = 500$		$t_0 = 510$		$t_0 = 520$		$t_0 = 530$		$t_0 = 550$	
	t_w^*	$\mathrm{EC}_1(t_0^*, t_w)$	t_w^*	$\mathrm{EC}_1(t_0^*, t_w)$	t_w^*	$\mathrm{EC}_1(t_0^*, t_w)$	t_w^*	$\mathrm{EC}_1(t_0^*, t_w)$	t_w^*	$\mathrm{EC}_1(t_0^*, t_w)$
0.20	387.50	3104.15	377.50	3117.76	367.50	3131.34	357.50	3144.89	337.50	3171.90
0.21	357.06	3115.88	347.06	3129.48	337.06	3143.06	327.06	3156.61	307.06	3183.62
0.22	326.83	3127.30	316.83	3140.90	306.83	3154.48	296.83	3168.03	276.83	3195.04
0.23	296.80	3138.41	286.80	3152.02	276.80	3165.60	266.80	3179.15	246.80	3206.16
0.24	266.95	3149.23	256.95	3162.84	246.95	3176.97	236.95	3189.97	216.95	3121.98
0.25	237.31	3165.75	227.31	3173.36	217.31	3186.94	207.31	3200.49	187.31	3227.50

Table 4.5. Optimal warranty period and its corresponding for Policy 2

c_t	$t_0 = 500$		$t_0 = 510$		$t_0 = 520$		$t_0 = 530$		$t_0 = 550$	
	t_w^*	$\mathrm{EC}_2(t_0^*, t_w)$	t_w^*	$\mathrm{EC}_2(t_0^*, t_w)$	t_w^*	$\mathrm{EC}_2(t_0^*, t_w)$	t_w^*	$\mathrm{EC}_2(t_0^*, t_w)$	t_w^*	$\mathrm{EC}_2(t_0^*, t_w)$
0.20	1000	763.774	1000	766.134	1000	768.493	1000	770.851	1000	775.566
0.21	1000	778.774	1000	783.234	1000	783.693	1000	786.151	1000	791.066
0.22	1000	793.774	1000	796.893	1000	798.893	1000	801.451	1000	806.566
0.23	1000	808.774	1000	811.434	1000	814.093	1000	816.751	1000	822.066
0.24	1000	823.774	1000	826.534	1000	829.293	1000	832.051	1000	837.566
0.25	1000	838.774	1000	841.634	1000	844.493	1000	847.351	1000	853.066

Table 4.6. Optimal release time and warranty period for Policy 1 and 2

c_t	Policy 1			Policy 2		
	t_0^{**}	t_w^{**}	$\mathrm{EC}_1(t_0^{**}, t_w^{**})$	t_0^{**}	t_w^{**}	$\mathrm{EC}_2(t_0^{**}, t_w^{**})$
0.20	237.48	882.37	678.89	375.55	864.32	645.67
0.21	108.22	1000	689.10	246.75	962.89	688.39
0.22	0	1000	697.37	113.47	1000	707.35
0.23	0	1000	702.48	0	1000	723.54
0.24	0	1000	706.12	0	1000	737.75
0.25	0	1000	703.31	0	1000	753.56

cost increases. In addition, when the warranty period t_w for fixed the testing cost per unit time c_t increases, the optimal release time decreases. Comparing Policy 1 and Policy 2, the optimal release time of Policy 1 is always earlier than that of Policy 2 and the total expected software cost of Policy 1 for optimal release time is always later than that of Policy 2; this is because the reliability increases and the residual faults in the software were effectively detected and removed in the operational phase during a given warranty period.

Table 4.4 and 4.5 examine the effect of the testing cost per unit time c_t and release time t_0 on the optimal warranty period t_w^* for the proposed Policy 1 and 2. It can be seen that when the testing cost per unit time c_t for fixed the release time t_0 increases, the optimal warranty period t_w^* shorten and the total expected software cost increases. In addition, when the release time t_0 for fixed the testing cost per unit time c_t increases, the optimal warranty period decreases. Comparing Policy 1 and Policy 2, the optimal warranty period of Policy 1 is always shorter than that of Policy 2 and the total expected software cost of Policy 1 for optimal warranty period is always greater than that of Policy 2; The reason is same in the case determination of the optimal release time.

Table 4.6 shows the effect of the testing cost per unit time c_t on the joint optimal policy of the release time and the warranty period. From Table 4.6, as c_t increases, the optimal release time t_0^{**} happens earlier. However, the optimal warranty period t_0^{**} cannot be seen. In addition, as the testing cost per unit time c_t increases, $\mathrm{EC}_1(t_0^{**}, t_w^{**})$, $\mathrm{EC}_2(t_0^{**}, t_w^{**})$ can be confirmed to also decrease.

5. Conclusions

This paper defined the time support by the project team ends in the operational phase as the warranty period and modeled the differences in the software execution environments in the testing phase and the operational phase by considering the software operational profile in the software reliability model based on NHPP. In addition, we considered the problem of determining the optimal testing period and the optimal warranty period that minimize the total expected software cost in terms of periodic software maintenance (*e.g.* update, service pack, *etc*). As a future problem, the same model can be built and analyzed even in other periodic software maintenance (*e.g.* upgrade, version up) policies.

References

Cha, J. H. (2000). On a better burn-in procedure, *Journal of Applied Probability*, **37**, 1099–1103.

Cha, J. H. (2001). Burn-in procedures for a generalized model, *Journal of Applied Probability*, **38**, 542–553.

Dohi, T., Okamura, H., Kaio, N. and Osaki, S. (2000). The age-dependent optimal warranty policy and its application to software maintenance contract, *Proceeding 5th International on Probability Safety Assessment Manage*, **4**, 2547–2552.

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

Jelinski, Z. and Moranda, P. B. (1972). *Software Reliability Research, Statistical Computer Performance Evaluation*, Freiberger, W. Ed., Academic Press, New York.

Kimura, M., Toyota, T. and Yamada, S. (1999). Economic analysis of software release problem with warranty cost and reliability requirement, *Reliability Engineering and System Safety*, **66**, 49–55.

Koch, H. S. and Kubat, P. (1983). Optimal release time of computer software, *IEEE Transaction on Software Engineering*, **9**, 323–327.

Mi, J. (1994). Burn-in and maintenance policies, *Advances in Applied Probability*, **26**, 207–221.

Musa, J. D. (1979). *Software Reliability Data*, Technique Report, Roma Air Development Center, USFA.

Okumoto, K. and Goel, A. L. (1980). Optimum release time for software system and based on reliability and cost criteria, *Journal of System and Software*, **1**, 315–318.

Pham, H. (1993). Software reliability assessment: Imperfect debugging and multiple failure types in software development, EG&G-RAMM-10737, Idaho National Engineering Laboratory.

Pham, H. and Zhang, X. (1999). A software cost model with warranty and risk costs, *IEEE Transactions on Computers*, **48**, 71–75.

Rinsaka, K. and Sandoh, H. (1999). A study on software maintenance service contracts, *Transactions of IEICE*, J82-A, 1819–1829.

Rinsaka, K. and Dohi, T. (2006). Optimal testing/maintenance design in a software development project, *Electronics and Communications in Japan*, **89**, 953–961.

Yamada, S. (1994). Optimal release problems with warranty period based on a software maintenance cost model, *Transactions on IPS Japan*, **35**, 2197–2202.