Turkish Journal of Electrical Engineering and Computer Sciences

Volume 15 | Number 3 Article 5

1-1-2007

Study of the Warranty Cost Model for Software Reliability with an Imperfect Debugging Phenomenon

D.R. PRINCE WILLIAMS

Turk J Elec Engin, VOL.15, NO.3 2007, © TÜBİTAK

Study of the Warranty Cost Model for Software Reliability with an Imperfect Debugging Phenomenon

D. R. PRINCE WILLIAMS
Department of Information Technology, Salalah College of Technology, Post Box: 608
Salalah-211, SULTANATE OF OMAN
e-mail: princeshree1@rediffmail.com

Abstract

Software reliability is one of the most important characteristics of software quality. Its measurement and management technologies employed during the software life-cycle are essential for producing and maintaining quality/reliable software systems. Herein, we discuss a modified approach to calculating the delivery cost of a software product, when warranty is to be provided, with an imperfect debugging phenomenon. Unlike existing cost models, here the strategy was to consider maintenance support given to the customer with an imperfect debugging phenomenon. The optimal release time can be calculated for various reliability levels by minimizing the cost. The delivery cost, reliability of the software system, and the optimal release time were calculated by using an imperfect debugging software reliability growth model. Numerical illustration supports the optimal release policies.

1. Introduction

Every software company is expected to release error-free computer software to its customers; however, complex computer software is commonly used nowadays. It has been increasingly difficult for developers to produce highly reliable software systems effectively. Thus, a necessity has arisen to control software, in terms of reliability, cost, and release time.

The life cycle (LC) of software includes a series of production activities, namely design, coding, testing, and operation/maintenance phases. In spite of great advancements in programming technology, the potential for errors due to human imperfection at every step are many. In other words, it is apt to say that software can never be made error-free.

There should be efficient management during the testing and maintenance phases of the software LC since delay in the release of software may cause a company significant financial loss. Hence, the software should reach customers at a desired level of reliability and with a minimum cost. Nonetheless, unfortunately software reaches customers before all faults are completely removed; thereby, the maintenance cost during the operational phase increases.

Hence, it is very important, in terms of software management, that we determine the optimal length of software testing, referred to as the optimum release time. Such a decision problem is known as an optimal software release problem. Many researchers have studied this decision problem and have presented different policies [2,4,6,15,23]. Okumoto et al. [10] studied the optimal release problem with simulated cost

and reliability requirements. Yamada et al. [21] implemented the optimal release problem with warranty cost and reliability requirement. Yang et al. [24] studied operational and testing reliability in software assessment models. Tal et al. [17] provided an optimal statistical testing policy for software reliability. Jain et al. [7] investigated hybrid warranty policy, the cost prediction of the total warranty reserve by including the effects of the time value of money. Popstojanova and Trivedi [13] studied the architecture-based approach to reliability assessment of software systems. Yamada et al. [22] implemented the optimal software release problems with LC distribution and the discount rate. Prince Williams et al. [14] studied the optimal software release problems with life-long warranty. Pham [12] studied a software cost model with imperfect debugging with random LC and penalty cost. Xie et al. [19] investigated the effect of imperfect debugging on software development cost, which, in turn, might affect the optimal software release time or operational budget. Bhaskar [1] developed the cost model based on criticality of the fault and the cost of its occurrence during different phases of development for an N-version programming scheme, a popular fault-tolerant architecture. Thus, from the above literature, different optimal release time polices are derived for various situations like:

- software development cost and reliability requirement;
- software development cost, imperfect debugging, and reliability requirement;
- software development cost, warranty period, warranty cost, and reliability requirement;
- software development cost, warranty period, warranty cost, test effort, and reliability requirement;
- software development cost, discount rate in developmental and operational cost, warranty period, warranty cost, test effort, and reliability requirement.

Nevertheless, it is very interesting to study the optimal release time of a software system with the combination of software development cost, discount rate in developmental and operational cost, warranty period, warranty cost, imperfect debugging, and reliability requirement. Hence, this cost model is realistic and practical.

We studied the optimal release time of software reliability with the combination of software development cost, warranty cost, discount rate, imperfect debugging, and reliability requirement. In section 1, we describe the applied software reliability growth model (SRGM) known as imperfect debugging SRGM. In section 2 we discuss the cost model. This cost model was developed for 2 situations: when reliability growth is constant and when reliability growth increases in the warranty period. In section 3, the optimal release time problem is discussed in detail. In section 4, we discuss optimal release policies that will help software vendors calculate the software release cost by taking into account the following:

- The software development cost with the discount rate;
- The software maintenance cost during the operational phase with the discount rate;
- Expected reliability level;
- The probability of imperfect debugging.

The optimal release time is determined by minimizing the total expected software cost and by satisfying the reliability requirement. In section 5, we discuss the optimal release time policies with the help of numerical illustration. In section 6, the conclusion is presented.

2. The Model

2.1. Non-Homogeneous Poisson Process (NHPP) Models and their use in Software Reliability

During software reliability testing and operation, software failures will randomly occur. Denoted by $N(t)$ the number of failures during $(0, t]$, $\{N(t), t \geq 0\}$ is then a stochastic process. Usually, $\{N(t), t \geq 0\}$ can be modeled by NHPP. The definition of NHPP is as follows:

A counting process $\{N(t), t \geq 0\}$ is said to be an NHPP with intensity function $\lambda(t)$ if it satisfies the following conditions:

(i) $N(0) = 0$,

(ii) $\{N(t), t \geq 0\}$ has independent increment,

(iii) $P_r[N(t+h) - N(t) = 0] = 1 - \lambda(t) + o(h)$,

(iv) $P_r[N(t+h) - N(t) = 1] = \lambda(t)h + o(h)$,

where $o(h)$ has the usual meaning $\lim_{h \to 0} \frac{o(h)}{h} = 0$.

The expectation of $N(t)$, denoted by $m(t)$, is called the mean value function of the process and its derivative, if it exists, will be equal to $\lambda(t)$. Clearly, $m(t)$ is a non-decreasing function of t and $m(0) = 0$. NHPP models play an important role in software reliability modeling. Generally, an NHPP software reliability model can be constructed by using an appropriate mean value function $m(t)$ or failure intensity function $\lambda(t)$. NHPP models are usually mathematically simple and practical. Given a set of failure time data, we can estimate the parameters in the model and the decision can be made as to when to stop the testing.

2.2. Software Reliability Growth Model (SRGM)

In the testing phase of software development and its operational phase, it is natural that we should apply probability and statistical theories to a software failure-occurrence or software fault-detection phenomenon for describing the stochastic behavior. Generally, the implemented software system is investigated to determine if the system can perform properly according to its specification requirements. Concretely, many test cases, which are generated prior to the testing phase from the specification requirements, are executed one by one in the (system) testing procedure. Each execution result confirms whether it is acceptable from the viewpoint of the specification. If the result is unacceptable, the software has at least one software fault, and then the fault must be eliminated. From the viewpoint of such a black-box testing process, software failure-occurrence and fault-detection phenomena can be treated as NHPP. These phenomena can be formulated by various mathematical models known as software reliability growth models (SRGM) [8,9,16].

2.3. Applied Model

In our model, NHPP is used to describe the time-dependent nature of the cumulative number of faults detected up to a specific testing time. If we consider the imperfect debugging software reliability growth

model [11,18], the mean value function and instantaneous function of the model are given by

$$m(t) = \frac{a}{p}(1 - e^{-bpt}), (a > 0, b > 0, 0 < p < 1), \tag{1}$$

$$\lambda(t) = abe^{-pbt} \tag{2}$$

where

a is the expected number of initial faults in a software;

b is the error detection rate per fault;

p is the probability of perfect debugging.

3. Warranty Cost Model

When the cost of software development is estimated, software vendors also have to consider the cost of after-sales support. This is known as the warranty cost. The computation of this warranty cost is dependent on the release time of the software.

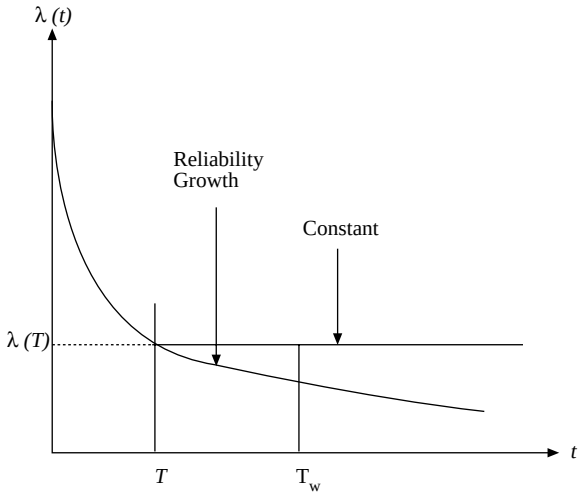

Figure 1. Software reliability growth for warranty period.

In this paper, we focus on the optimal release time problem for developmental cost and warranty cost with the discount rate in the imperfect debugging phenomenon. The total expected software product cost, including maintenance cost, can be formulated as [5]

$$C(T) = c_0 + c_t \int_0^T e^{-\alpha t} dt + C_w(T). \tag{3}$$

where

c_0 is the initial testing cost, which is the bare minimum requirement;

c_t is the testing cost per unit time;

T is the software release time;

α is the discount rate of the total software cost;

$C_w(T)$ is the maintenance cost during the warranty period.

There are 2 cases in terms of the behavior of $C_w(T)$ (Figure 1).

Case 1. Let us assume that during the warranty period software reliability growth does not occur after the testing phase (assuming that we correct only minor errors that will not affect the reliability of the software). Then $C_w(T)$ is defined as

$$C_w(T) = c_w \int_T^{T+Tw} \lambda(T) e^{-\alpha T} dt \tag{4}$$

where

T_w is the warranty period of the software and

c_w is the maintenance cost per fault during the warranty period.

Case 2. Let us assume that during the warranty period software reliability growth occurs, even after the testing phase (assuming that we correct only major errors that will improve the reliability of the software). Then $C_w(T)$ is defined as

$$C_w(T) = c_w \int_T^{T+Tw} \lambda(t) e^{-\alpha t} dt \tag{5}$$

4. Warranty Cost Model with Reliability Constraint

The software reliability of the NHPP model [3,5,9] is defined as the probability that a software failure will not occur during the testing time interval $(T, T+x]$ and is mathematically stated as

$$R(x/T) = \exp[-\{m(T+x) - m(T)\}] \tag{6}$$

substituting Eq. (1) in Eq. (6), we get

$$R(x/T) = \exp[-e^{-bpT} m(x)] \tag{7}$$

Let R_0 be the desired level of reliability. Then the optimal release problem [20] is formulated as

$$\left.\begin{array}{ll} \text{Minimize} & C(T) \\ \text{Subject to} & R(x/T) \geq R_0 \end{array}\right\} \tag{8}$$

4.1. Optimal Software Release Policies: Cost Minimization

Having defined the problem, we now derive the optimal release time policies by minimizing the total expected software cost $C(T)$.

Optimal Release Policy 1 Based on $Case(1)$ Case 1 the total expected software product cost is given by

$$C(T) = c_0 + c_t \int_0^T e^{-\alpha t} dt + c_w \int_T^{T+T_w} \lambda(T) e^{-\alpha T} dt \tag{9}$$

Replacing $\lambda(T)$ with abe^{-pbT} in Eq. (9), we get

$$C(T) = c_0 + c_t\left(\frac{1-e^{-\alpha t}}{\alpha}\right) + c_w T_w abe^{-(\alpha+bp)T} \tag{10}$$

Differentiating Eq. (10) with respect to T and equating to zero, we get

$$T = T_1 = \frac{1}{bp}\ln\left(\frac{ab(\alpha+bp)T_w c_w}{c_t}\right) \tag{11}$$

The second derivative of $C(T)$ is greater than zero, that is $\frac{d^2(C(T))}{dT_1^2} > 0$.

Hence, $C(T)$ has a minimum value. **Optimum Release Policy 1** can now be stated as

$$\boldsymbol{P1.1}\, T^* = T_1 \text{ when } \lambda(0) > \lambda(T_1) \tag{12}$$

$$\boldsymbol{P1.2}\ T^* = 0 \text{ when } \lambda(0) \leq \lambda(T_1) \tag{13}$$

where T^* is the optimal release time of the software.

Optimal Release Policy 2 Based on Case 2 the total expected software maintenance cost is given by

$$C(T) = c_0 + c_t\int_0^T e^{-\alpha t}dt + c_w\int_T^{T_w}\lambda(t)e^{-\alpha t}dt \tag{14}$$

where $\lambda(t) = abe^{-pbt}$

$$C(T) = c_0 + c_t\left(\frac{1-e^{-\alpha T}}{\alpha}\right) + \frac{c_w ab\left(1-e^{-(\alpha+bp)T_w}\right)e^{-(\alpha+bp)T}}{(\alpha+bp)} \tag{15}$$

Differentiating Eq. (15) with respect to T and equating to zero, we get

$$T = T_2 = \frac{1}{bp}\ln\left(\frac{ab\left(1-e^{-\alpha T_w}\right)c_w}{c_t}\right) \tag{16}$$

Again, the second derivative of $C(T)$ is greater than zero, that is $\frac{d^2(C(T))}{dT_2^2} > 0$.

Hence, $C(T)$ has a minimum value. **Optimum Release Policy 2** in this case, can be stated as follows:

$$\boldsymbol{P2.1}\, T^* = T_2 \text{ when } \lambda(0) > \lambda(T_2) \tag{17}$$

$$\boldsymbol{P2.2}\, T^* = 0 \text{ when } \lambda(0) \leq \lambda(T_2) \tag{18}$$

4.2. Optimal Software Release Policies: Cost and Reliability Requirement

These policies are derived by considering both the minimization of software cost as well as the required level of reliability. Let T_R be the optimum release time, with respect to T, satisfying the relation $R(x/T) = R_0$. By applying the relation $R(x/T) = R_0$ in Eq. (7), we obtain T_R as

$$T_R = \frac{1}{bp}\left\{\ln m(x) - \ln\ln\left(\frac{1}{R_0}\right)\right\} \tag{19}$$

Having obtained T_R, the optimal release policies, including minimization of the total expected software cost, can be derived for the 2 cases identified earlier.

Case 1. Optimum Release Policy 3.

$$\textbf{\textit{P3.1}}\ \text{If } \lambda(0) > \lambda(T_1) \text{ and } R(x/0) < R_0,\ \text{then } T^* = \max\{T_1, T_R\} \tag{20}$$

$$\textbf{\textit{P3.2}}\ \text{If } \lambda(0) > \lambda(T_1) \text{ and } R(x/0) \geq R_0,\ \text{then } T^* = T_1 \tag{21}$$

$$\textbf{\textit{P3.3}}\ \text{If } \lambda(0) \leq \lambda(T_1) \text{ and } R(x/0) < R_0,\ \text{then } T^* = T_R \tag{22}$$

$$\textbf{\textit{P3.4}}\ \text{If } \lambda(0) \leq \lambda(T_1) \text{ and } R(x/0) \geq R_0,\ \text{then } T^* = 0 \tag{23}$$

Case 2. Optimum Release Policy 4.

$$\textbf{\textit{P4.1}}\ \text{If } \lambda(0) > \lambda(T_2) \text{ and } R(x/0) < R_0,\ \text{then } T^* = \max\{T_2, T_R\} \tag{24}$$

$$\textbf{\textit{P4.2}}\ \text{If } \lambda(0) > \lambda(T_2) \text{ and } R(x/0) \geq R_0,\ \text{then } T^* = T_2 \tag{25}$$

$$\textbf{\textit{P4.3}}\ \text{If } \lambda(0) \leq \lambda(T_2) \text{ and } R(x/0) < R_0,\ \text{then } T^* = T_R \tag{26}$$

$$\textbf{\textit{P4.4}}\ \text{If } \lambda(0) \leq \lambda(T_2) \text{ and } R(x/0) \geq R_0,\ \text{then } T^* = 0 \tag{27}$$

5. Numerical Illustration

We assume the parameter values for the model as

$c_0 = 1000, c_w = 20, a = 1000, b = 0.05, p = 0.9,$ and $\alpha = 0.001$.

The optimal release times are then calculated for different values of unit testing cost (c_t) and warranty period (T_w), using the above parameter values, for Policies 1-4, defined earlier. These have been tabulated and shown in Tables 1-4, respectively.

Table 1. Optimal Release Policy 1 ($T^* = T_1$).

$T_w \setminus c_t$	1	5	10	20	30	40
1	68.92	39.95	27.47	14.99	7.69	2.52
5	97.89	68.92	56.44	43.96	36.66	31.49
10	110.36	81.39	68.92	56.44	49.14	43.96
20	122.84	93.87	81.39	68.92	61.62	56.44
30	130.14	101.17	88.69	76.21	68.92	63.74
40	135.32	106.35	93.87	81.39	74.09	68.92
50	139.33	110.36	97.89	85.41	78.11	72.93
100	151.81	122.84	110.36	97.89	90.59	85.41

Table 2. Optimal Release Policy 2($T^* = T_2$).

$T_w \setminus c_t$	1	5	10	20	30	40
1	84.57	48.81	33.40	18.00	8.99	2.60
5	118.34	82.57	67.17	51.77	42.76	36.36
10	131.33	95.57	80.17	64.76	55.75	49.36
20	142.21	106.44	91.04	75.64	66.63	60.23
30	147.07	111.30	95.90	80.49	71.48	65.09
40	149.66	113.90	98.49	83.09	74.08	67.69
50	151.16	115.39	99.99	84.59	75.58	69.18
100	153.28	117.52	102.11	86.71	77.70	71.31

From Tables 1 and 2 it can be seen that when the unit testing cost (c_t) and warranty time (T_w) increase, the optimal release time decreases. Comparing Policy 1 (Table 1) and Policy 2 (Table 2), the optimal release time of Policy 2 is always greater than that of Policy 1; this is because reliability increases during the warranty period. However, for a longer warranty period the optimal release time of Policy 1 is always greater than that of Policy 2; this is because the learning rate of debugging increases. The optimal release times of Policy 3 and Policy 4 are given in Tables 3 and 4, respectively, from which it can be seen that when the unit testing cost (c_t) and warranty time (T_w) increase the optimal release time remains constant, since for all cases the optimal release time ($T_R = 166.62$) is at the maximum compared to T_1 and T_2.

Table 3. Optimal Release Policy 3.

$T_w \setminus c_t$	1	5	10	20	30	40
1	166.62	166.62	166.62	166.62	166.62	166.62
5	166.62	166.62	166.62	166.62	166.62	166.62
10	166.62	166.62	166.62	166.62	166.62	166.62
20	166.62	166.62	166.62	166.62	166.62	166.62
30	166.62	166.62	166.62	166.62	166.62	166.62
40	166.62	166.62	166.62	166.62	166.62	166.62
50	166.62	166.62	166.62	166.62	166.62	166.62
100	166.62	166.62	166.62	166.62	166.62	166.62

Next, we also calculated optimal release time (T_R)for the various levels of reliability (R_0) and operational period lengths (x) from Eq. (14).

Assuming $a = 1000, p = 0.9, \alpha = 0.001$ and b = 0.05, we tabulated the results as follows:

Table 4. Optimal Release Policy 4.

$T_w \setminus c_t$	1	5	10	20	30	40
1	166.62	166.62	166.62	166.62	166.62	166.62
5	166.62	166.62	166.62	166.62	166.62	166.62
10	166.62	166.62	166.62	166.62	166.62	166.62
20	166.62	166.62	166.62	166.62	166.62	166.62
30	166.62	166.62	166.62	166.62	166.62	166.62
40	166.62	166.62	166.62	166.62	166.62	166.62
50	166.62	166.62	166.62	166.62	166.62	166.62
100	166.62	166.62	166.62	166.62	166.62	166.62

Table 5. Optimal release time (T_R) for varies values of R_0 and x.

R0 / x	0.1	0.2	0.4	0.5	0.7	0.8	0.9
1	67.90	75.86	88.38	94.58	109.35	119.77	136.44
2	82.81	90.77	103.29	109.49	124.25	134.68	151.35
5	101.71	109.67	122.19	128.39	143.16	153.58	170.25
10	114.76	122.71	135.23	141.43	156.20	166.62	183.30
20	125.72	133.68	146.19	152.40	167.16	177.58	194.26

Table 5 shows the optimal release time (T_R) satisfying Eq. (14) for various values of the software reliability requirement (R_0) and the operational period (x). The values in Table 5 indicate that the optimal release time increases as the operational period increases.

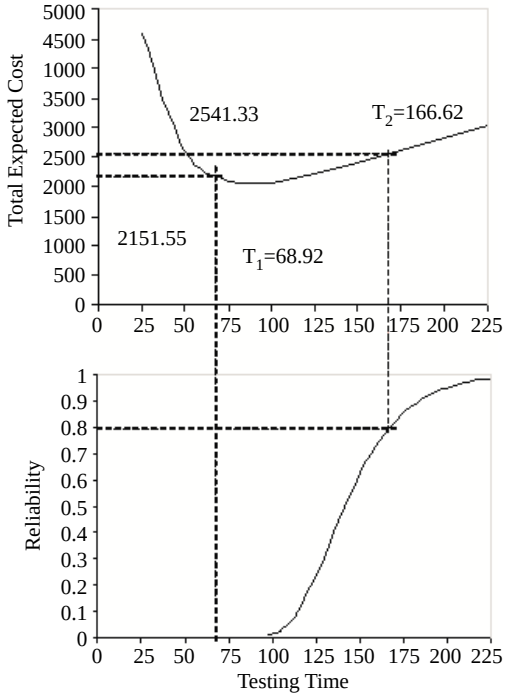

Figure 2. Optimal release time problem.

For Optimal Release Policies 3 and 4, the optimal release time remains constant, that is 166.62, for different warranty periods, since for the values of $c_0 = 1000, c_w = 20, T_w = 10, c_t = 10, a = 1000, b = 0.05, p = 0.9, R_0 = 0.8$, and $\alpha = 0.001$, the optimal release time is $T^* = \max\{T_1, T_R\} = \max\{68.92, 166.62\} = T_R = 166.62$ and $T^* = \max\{T_2, T_R\} = \max\{80.17, 166.62\} = T_R = 166.62$ (Tables 1, 2, and 5). Figure 2 illustrates the optimal release time with the reliability requirement (R_0) for Optimal Release Policy 3. In the case of $T_w = 10, c_t = 10$, we derive $T_1 = 68.92$ (Table 1). We also obtained $T_R = 166.62$ (Table 5) for the values of $R_0 = 0.8$ and $x = 10$, since $\lambda(0) = 50 > 2 = \lambda(68.92) = \lambda(T_1)$ and $R(x = 10/0) = 0 < 0.8 = R_0$. Therefore, using Optimal Release Policy 3 (P3.1) of Eq. (20), (Figure 2) the optimal release time is

$$T^* = \max\{T_1, T_R\} = \max\{68.92, 166.62\} = 166.62.$$

5.1. Sensitivity Analysis

For the sake of sensitivity analysis of this cost model in imperfect debugging, we assume $c_0 = 1000, c_w = 20, p = 0.9, \alpha = 0.001, R_0 = 0.8$. Tables 6 and 8 show the optimal release times for different values of a and b for optimal release policies 1 and 2, respectively.

Table 6. Optimal release rime of policy 1 for different values of a and b.

a / b	0.03	0.05	0.08	0.2	0.5	0.8
2000	63.85	56.44	45.76	26.51	13.90	9.74
1500	55.22	51.26	42.52	25.22	13.38	9.42
1000	43.05	43.96	37.96	23.39	12.65	8.96
900	39.89	42.07	36.77	22.92	12.46	8.84
500	22.26	31.49	30.16	20.27	11.40	8.18
400	15.56	27.47	27.65	19.27	11.00	7.93
300	6.93	22.29	24.41	17.98	10.48	7.61

Table 7. Optimal release time of policy 1 for different values of a and p.

a / p	0.3	0.5	0.7	0.8	0.9
2000	12.48	25.65	40.47	48.33	56.44
1500	10.75	22.77	36.44	43.72	51.26
1000	8.32	18.72	30.76	37.24	43.96
900	7.69	17.66	29.29	35.55	42.07
500	4.16	11.79	21.06	26.15	31.49
400	2.82	9.56	17.93	22.58	27.47
300	1.09	6.68	13.91	17.97	22.29

Table 8. Optimal release time of policy 2 for different values of a and p.

a / b	0.03	0.05	0.08	0.2	0.5	0.8
2000	76.26	67.17	54.03	30.41	15.10	10.21
1500	65.61	60.78	50.04	28.81	14.47	9.81
1000	50.59	51.77	44.41	26.55	13.56	9.25
900	46.69	49.43	42.94	25.97	13.33	9.10
500	24.92	36.36	34.78	22.70	12.02	8.28
400	16.65	31.41	31.68	21.46	11.53	7.97
300	6.00	25.01	27.68	19.87	10.89	7.57

Table 9. Optimal release time of policy 2 for different values of a and b.

a / p	0.3	0.5	0.7	0.8	0.9
2000	135.98	100.03	80.05	72.99	67.17
1500	116.80	88.52	71.83	65.80	60.78
1000	89.77	72.30	60.24	55.66	51.77
900	82.75	68.09	57.23	53.03	49.43
500	43.56	44.57	40.44	38.33	36.36
400	28.68	35.65	34.07	32.76	31.41
300	9.51	24.14	25.85	25.56	25.01

In particular, from Table 6 it is clear that if the expected total number of errors (a) decreases and the error detection rate (b) increases, then the optimal release time ($T^* = T_1$) decreases, and from Table 8 a similar result is obtained for the optimal release time ($T^* = T_2$). Assuming $c_0 = 1000, c_w = 20, b = 0.05, \alpha = 0.001, R_0 = 0.8$, Tables 7 and 9 show the optimal release time for different values of a and p for optimal release policies 1 and 2, respectively. In particular, from Tables 7 and 9 it is clear that if the expected total number of errors (a) decreases and the probability of perfect debugging (p) increases then the optimal release time decreases (for both $T^* = T_1$ and $T^* = T_2$). These analyses help the software developer to choose the appropriate optimal release time for their requirements.

6. Conclusion

In this paper, we derived optimal release time policies to predict the optimal release time of software in imperfect debugging phenomena with the warranty period. Hence, it would help a software vendor to calculate the total software product cost for the warranty period in imperfect debugging phenomena. Moreover, the cost model accommodates the following 2 situations in imperfect debugging phenomena. The first is that the warranty is provided to only retain the reliability level promised at the time of software release. The second is that the warranty is provided to increase the reliability level from that stated at the time of software release. The sensitivity analysis of optimal release policies was also discussed. The numerical example supports the derived optimal release policies.

7. Acknowledgment

The author is grateful to the referee's valuable constructive suggestions for the improvement of this paper.

References

[1] T. Bhaskar, U.D. Kumar, "A cost model for N-version programming with imperfect debugging", Journal of the Operational Research Society, Vol. 57 (8), pp. 986-994, 2006.

[2] J.K. Chaar, M.J. Halliday, T.S. Bhandari, R.Chillarege, "In-process evaluation for software inspection and test", IEEE Trans. Softw. Engg, Vol. 19 (10)pp.1055-1070, 1993.

[3] B.C. Cho, K.S. Park, "An optimal time for software testing under the user's requirement of failure-free demonstration before release", IEICE Trans. Fundamentals, Vol. E77-A (3), pp. 563-570, 1994.

[4] E.H. Foreman, N.D. Singpurwalla, "Optimal time intervals for testing hypotheses on computer software errors", IEEE Trans. Reliability, Vol. 28 (3), pp. 250-253, 1979.

[5] M. Kimura, T. Toyota, S. Yamada, "Economic analysis of software release problems with warranty cost and reliability requirement", Reliability Engineering and System Safety, Vol 66, pp. 49-55, 1999.

[6] M. Kimura, S. Yamada, "Optimal software policies with random life-cycle and delivery delay", In: Procedings of the 2^{nd} ISSAT Int. Conf. Reliability and Quality in Design, pp. 215-219, 1995.

[7] M. Jain, B.R. Handa, "Cost analysis for repairable units under hybrid warranty. Recent developments in Operational Research", (ed. Manju Lata Agarwal & Kanwar Sen), New Delhi, Narosa Publishing House, pp. 149-165, 2001.

[8] Y.K. Malaiya, P.K. Srimani (Eds.), Software Reliability Models: Theoretical Developments, Evaluation and Applications, Los Alamitos, IEEE Computer Society Press, 1990.

[9] J.D. Musa, A. Tannino, K. Okumoto, Software reliability: measurement, prediction, and application, New York, McGraw-Hill, 1987.

[10] K. Okumoto, A.L. Goel, "Optimum release time for software system based on reliability and cost criteria", J. System and Software, Vol. 14, pp. 315-318, 1980.

[11] H. Pham, (Editor), Software Reliability and Testing, Los Alamitos, IEEE Computer Society Press, 1990.

[12] H. Pham, "A software cost model with imperfect debugging, random life cycle and penalty cost", Int. J. System Sci, Vol. 27, pp. 455-463, 2003.

[13] K.G. Popstojanova, K.S. Trivedi, "Architecture-based approach to reliability assessment of software systems", Performance Evaluation, Vol. 45, pp. 179-204, 2001.

[14] D.R. Prince Williams, P. Vivekanandan, "Life-Time Warranty Cost Model for Software Reliability with Discount Rate", Journal of Computer Science, Vol. 1, pp. 53-59, 2005.

[15] N.E. Rallis, Z.F. Lansdowne, "Reliability estimation for a software system with sequential independent reviews" IEEE Transactions on Software Engineering, Vol. 27 (12), pp. 1057-1061, 2001.

[16] M.L. Shooman, Software engineering: design, reliability, and management, New York, McGraw-Hill, 1983.

[17] U. Tal, C. McCollin, A. Ben Dell, "An optimal statistical testing policy for software reliability", Demonstration of safety critical systems, Vol 137 (3), pp. 544-557, 2002.

[18] M. Xie, Software reliability modeling, Singapore: World Scientific, 1991.

[19] M. Xie, B. Yang, "A study of the effect of imperfect Debugging on software development cost model", IEEE Trans. on Software Engineering, Vol. 29(5), pp. 471-473, 2003.

[20] S. Yamada, "Software quality/reliability measurement and assessment: software reliability growth models and data analysis", J. Info. Process, Vol. 14 (3), pp. 254-266, 1991.

[21] S. Yamada, "Optimal release problems with warranty period based on a software maintenance cost model", Trans. IPS Japan, Vol. 35 (9), pp. 2197-2202, 1994.

[22] S. Yamada, M. Kimura, E. Terane, S. Osaki, " Optimal software release problems with life-cycle distribution and discount rate", Trans. IPS Japan, Vol. 34(5), pp. 1188-1197, 1993 (in Japanese).

[23] S. Yamada, S. Osaki , "Optimal software release policies with simultaneous cost and reliability requirements", Eur. J. Oper. Res, Vol. 31 (1), pp. 46-51, 1987.

[24] B. Yang, M. Xie, "A study of operational and testing reliability in software reliability analysis", Reliability Engineering and System Safety, Vol. 70, pp. 323-329, 2000.