Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2013) 21: 234 – 245

doi:10.3906/elk-1105-5

# Optimal release policies for a software system with warranty cost and change-point phenomenon

**D. R. Prince WILLIAMS***
Department of Information Technology, College of Applied Sciences-Sohar,
(Ministry of Higher Education), PB: 135, Sohar-311, Oman

**Received:** 03.05.2011 • **Accepted:** 19.10.2011 • **Published Online:** 27.12.2012 • **Printed:** 28.01.2013

**Abstract:** Determining the software quality and release time of the software is an important role of software reliability. Software reliability growth models (SRGMs) are applications of software reliability. The change-point is defined as the prominent change in the software testing time, in connection with the quality of a software error occurrence or error-detection phenomenon. As a result, the software reliability growth process, based on the change-point, affects the precision of the software reliability prediction, based on the SRGMs. To find the accurate total expected delivery cost with a suitable warranty period of the software system, a new cost model for the software system with the warranty and change-point phenomenon is proposed in this paper. The entire expected delivery cost and the reliability of the software system is calculated using the change-point SRGM. The optimal release time is calculated by reducing the overall estimated delivery cost for various desired reliability levels. Based on the proposed warranty cost model and the reliability of the software system, we have derived some optimal release policies. Numerical illustrations and interconnected discussed data are itemized. From the testing outcome, we get software release policies that provide a comprehensive analysis of software based on the total expected delivery cost with a suitable warranty period, desired level of reliability, and change-point. Moreover, these policies will be helpful for project managers to decide where to stop the testing for customer release at the exact time with a suitable warranty period.

**Key words:** Software reliability, optimal release policies, change-point

## 1. Introduction

Computer systems play a vital and indispensable role in our day-to-day lives. Computer hardware as well as the software forms an entire system. Most of the system can function with the help of a software program. In the 21st century, we would rarely find any company or service institute functioning exclusively without the assistance of an implanted software system. This requirement of humanity for software systems has made it essential to yield a great number of reliable software programs. The level of reliability constraints is very high for the real-time safety-critical control system software. There are ample realistic illustrations where software errors have caused striking failures leading to disasters to life and economy.

Recently, a large number of software developers are using an effectual, controlled, and procedurally designed development of the software process to yield good-quality software. The process of identifying errors and ridding the software of errors is a very difficult and expensive process. The cost spent on correcting the faults after the release time results in a higher development cost. In the software testing stage, the testing group aims to expose and consequently remove the majority of the software faults. Nevertheless, developing error-free

*Correspondence: princeshree1@gmail.com

software is not possible due to an inadequacy in cost and time. The delay in the software release may cause it to be outdated or may cause customer dissatisfaction. There is a possibility of delayed released software to be antiquated in the market. Hence, every software manager should maintain the balance between the desired level of reliability and the software release time.

In the software testing and software operational stage, one can predict the software reliability using a suitable software reliability growth model (SRGM). During the various stages of the software development process, management should use different optimization techniques. The optimal software release time problem depends on various components. The optimal testing time is dependent on a variety of factors, such as the skill of the software developing and testing team, desired level of reliability, warranty period, etc. Many researchers have studied and developed optimal software release time problems based on different real-life situations. To study the different software release time problems based on different environments, see [1–10]. The first unconstrained optimal release time problem was formulated by Okumoto et al. in [11]. The constrained optimal release time problem optimizes the cost minimization and reliability maximization, developed by Yamada et al. [12]. The bicriterion optimal release time problem was studied and discussed by Kapur et al. [13]. In [14], Jain et al. discussed in detail the optimal release time problem, based on the warranty time and warranty cost. Bhaskar and Kumar [15] studied the 3-version program optimal release policy and discussed various aspects of the release time.

This paper is systematized as follows. A concise review of an applied SRGM with the change-point for software reliability is provided in Section 2. In Section 3, we have discussed the warranty cost model for the change-point software reliability. In Section 4, we have presented the optimal release policies incorporated with the warranty period, warranty cost, software development cost with a discount rate, software maintenance cost during the operational phase with a discount rate, and the change-point. In Section 5, we have discussed optimal release time policies with numerical examples. We conclude the paper in Section 6.

## 2. The model

### 2.1. A change-point nonhomogeneous Poisson process model

We can model the total number of software errors up to the period of time $t$ as a discrete random process $\{N(t), t \geq 0\}$. This discrete random process, $N(t)$, is a nonhomogeneous Poisson process (NHPP) with mean value function $m(t)$. The mean value function of the discrete random process $N(t)$ is the expected total number of errors by given time $t$. In [16], the total number of software errors during the disjoint time intervals is modeled as a NHPP with the assumptions that the errors are independent and the software failure intensity, $\lambda(t)$, is proportional to the residual fault content. The contribution of the NHPP model in the software reliability modeling plays a very important role. Normally, a NHPP software reliability model is developed choosing a suitable mean value function, $m(t)$, or a failure intensity function, $\lambda(t)$. The mathematically elegant and practical NHPP models are very easy to apply for the given failure dataset to predict the parameters of the SRGM model and when to terminate the software system testing.

In the literature, most of the SRGMs [2,7,10] have been developed with the assumption that the error detection function is a constant or a decreasing function, and the failure intensity is a continuous time function. For illustration, the constant fault detection rate was given by Goel and Okumoto [16]. Furthermore, a development to provide the debugging method with the learning phenomenon, represented as an increasing fault detection rate function, was given as a modified G-O model by Yamada et al. [17]. The above 2 models were developed based on the continuous failure intensity function. As an earlier argument, the testing strategy

and resource allocation can be dominated by the fault detection rate. In the process of software testing, there is a chance that for some time moment $s$ the fault detection rate may change. This change in time $s$ is called the change-point. To provide more realistic SRGMs, we should incorporate this change-point in the development of the SRGM.

### 2.2. Applied change-point NHPP model

In the change-point testing environment, we intended to create the following cost model for the software system using a NHPP SRGM. In general, the occurrence of the error detection rate might not be same and can be tailored with regard to the changes in the resource allocation, strategy, or testing environment. The error detection rate of this change-point NHPP (NHPP-CP) at time $t$is given by [18]:

$$b(t) = \frac{\frac{dm(t)}{dt}}{(a - m(t))} = \begin{cases} b_1 & \text{for} \quad 0 \le t \le \tau, \\ b_2 & \text{for} \quad t > \tau. \end{cases}, \tag{1}$$

where $\tau$ is the change-point, $a > 0$ is the expected number of software errors at the beginning of the test, and $b_1, b_2 > 0$ are the error detection rates before and after the change-point, respectively.

If $b_1 = b_2 = b$ in Eq. (1), the NHPP-CP model is the same as the G-O model in [16].

Solving Eq. (1) for $m(t)$ gives the following.

$$m(t) = \begin{cases} a\left(1 - e^{-b_1 t}\right), & \text{for} \quad 0 \le t \le \tau, \\ a\left(1 - e^{-b_1 \tau - b_2 (t-\tau)}\right), & \text{for} \quad t > \tau. \end{cases} \tag{2}$$

$$\lambda(t) = \frac{dm(t)}{dt} = \begin{cases} ab_1 e^{-b_1 t} & \text{for} \quad 0 \le t \le \tau, \\ ab_2 e^{-b_1 \tau - b_2 (t-\tau)} & \text{for} \quad > \tau. \end{cases} \tag{3}$$

## 3. Warranty cost model

In reality, software reliability estimation is generally not sufficient because there are 2 important risk factors: the stopping time of the software for release to the customer and the expected total development cost of the software. Hence, the period of testing and the methods of testing used are the important factors to judge the quality of the software system. Normally, to provide a more reliable and fault-free software system, we consider a longer testing time. A longer testing time will result in a higher total development cost of the software, whereas a shorter testing time will result in a lower total development cost of the software, which causes a greater risk of the customer purchasing defective software. This results in a higher cost in the operational phase. According to the software development company's information, the cost of debugging a fault during the operational phase is about 20 times higher than that of the testing phase. Hence, to reduce the expected total development cost of the software, it is important to predict the optimal release time policies for the stopping time of the software. Many software companies will also have to consider the after-sales support. This will impact the increase in development cost. This cost is known as the warranty cost. The release time of the software is very important in the computation of the warranty cost.

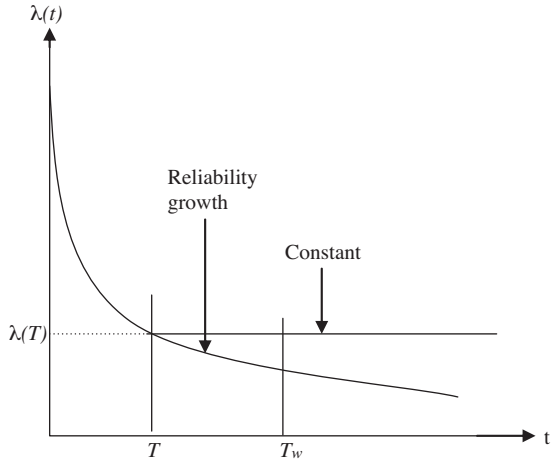


**Figure 1.** Software reliability growth for the warranty period.

In this paper, we developed a maintenance cost model for the formulation of the optimal release time problem. The warranty period is also incorporated in this maintenance cost model. The following equation represents the total expected software maintenance cost $C(T)$ as given in [19]:

$$C(T) = c_0 + c_t \int_0^T e^{-\alpha t} dt + C_w(T), \tag{4}$$

where $c_0$ is the minimum requirement in the initial testing cost, $c_t$ is the testing cost per unit time, $T$ is the software release time, $\alpha$ is the discount rate of the total software cost, and $C_w(T)$ is the maintenance cost during the warranty period.

Assume that the growth of the software reliability exists after the testing stage by correcting the major errors (see Figure 1). The same growth will also happen in the warranty period. Next, $C_w(T)$ is defined as:

$$C_w(T) = c_w \begin{cases} \int_T^{T+T_w} ab_1 e^{-b_1 t} dt & \text{for} \quad 0 \le t \le \tau, \\ \int_T^{T+T_w} ab_2 e^{-b_2 \tau - b_1 (t-\tau)} dt & \text{for} \quad t > \tau. \end{cases} \tag{5}$$

Here, $T_w$ is the software warranty period and $c_w$ is the maintenance cost per fault during the warranty period.

Thus:

$$C(T) = c_0 + c_t \int_0^T e^{-\alpha t} dt + c_w \begin{cases} \int_T^{T+T_w} ab_1 e^{-b_1 t} dt & \text{for} \quad 0 \le t \le \tau, \\ \int_T^{T+T_w} ab_2 e^{-b_2 \tau - b_1 (t-\tau)} dt & \text{for} \quad t > \tau. \end{cases} \tag{6}$$

$$C(T) = c_0 + c_t \left( \frac{1 - e^{-\alpha t}}{\alpha} \right) - a c_w \begin{cases} b_1 \left( \frac{e^{-(b_1+\alpha)(T+T_w)} - e^{-(b_1+\alpha)(T)}}{(b_1+\alpha)} \right) & \text{for} \quad 0 \le T \le \tau, \\ b_2 e^{-(b_1-b_2)\tau} \left( \frac{e^{-(b_2+\alpha)(T+T_w)} - e^{-(b_2+\alpha)(T)}}{(b_2+\alpha)} \right) & \text{for} \quad T > \tau. \end{cases} \tag{7}$$

## 4. Warranty cost model with the reliability constraint

In this section, to achieve the desired level of software reliability, we developed different levels of optimal release time polices. These optimal release time policies help the software manger to make a decision about when to stop the testing time and when to release the software to customers with respect to their desired level of reliability. From the SRGM [16,20–22], the software reliability function can be defined as the probability that a software failure does not occur during the time interval $(T, T+x]$ after the total testing time $T$, i.e. the release time. The software reliability function is given as follows:

$$R(x/T) = \exp[-\{m(T+x) - m(T)\}]. \tag{8}$$

Substituting Eq. (2) into Eq. (8), we get:

$$R\left({}^{x}/_{T}\right) = \begin{cases} \exp\left\{a\left(e^{-b_1(T+x)} - e^{-b_1 T}\right)\right\} & \text{for} \quad 0 \le T \le \tau, \\ \exp\left\{a\left(e^{-b_1\tau - b_2(T+x-\tau)} - e^{-b_1\tau - b_2(T-\tau)}\right)\right\} & \text{for} \quad T > \tau. \end{cases} \tag{9}$$

That is:

$$R\left({}^{x}/_{T}\right) = \begin{cases} \exp\left\{-e^{-b_1 T} m_1(x)\right\} & \text{for} \quad 0 \le T \le \tau, \\ \exp\left\{-e^{-b_2 T + (b_2 - b_1)\tau} m_2(x)\right\} & \text{for} \quad T > \tau. \end{cases} \tag{10}$$

Here, $m_1(x) = a\left(1 - e^{-b_1 x}\right)$ and $m_2(x) = a\left(1 - e^{-b_2 x}\right)$.

Let $R_0$ be the desired level of reliability. Hence, the optimal release time problem [21] is formulated as:

Minimize $C(T)$

$$\text{Subject to} \qquad R(x/T) \ge R_0. \tag{11}$$

### 4.1. Optimal software release policies: cost minimization

Now we define and derive the problem of the optimal release time policies by minimizing the total expected software development cost, $C(T)$.

#### Optimal release policy 1

The total expected software product cost is given by:

$$C(T) = c_0 + c_t\left(\frac{1 - e^{-\alpha t}}{\alpha}\right) - a c_w \begin{cases} b_1\left(\frac{e^{-(b_1+\alpha)(T+T_w)} - e^{-(b_1+\alpha)T}}{(b_1+\alpha)}\right) & \text{for} \quad 0 \le T \le \tau, \\ b_2 e^{-(b_1-b_2)\tau}\left(\frac{e^{-(b_2+\alpha)(T+T_w)} - e^{-(b_2+\alpha)T}}{(b_2+\alpha)}\right) & \text{for} \quad T > \tau. \end{cases} \tag{12}$$

Differentiating Eq. (6) with respect to $T$ and equating to 0, we get:

$$T = T_1 = \frac{1}{b_1}\ln\left(\frac{c_w a b_1\left(1 - e^{-(b_1+\alpha)T_w}\right)}{c_t}\right) \tag{13}$$

and

$$T = T_2 = \frac{1}{b_2}\ln\left(\frac{c_w a b_2 e^{-(b_1-b_2)}\left(1 - e^{-(b_1+\alpha)T_w}\right)}{c_t}\right). \tag{14}$$

The second derivative of $C(T)$ is greater than 0; 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:

***P1.1*** $T^* = T_1$ *when* $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ *and* $\lambda(\tau) < \lambda(T_2)$.

***P1.2*** $T^* = T_2$ *when* $\lambda(0) < \lambda(T_1) \leq \lambda(\tau)$ *and* $\lambda(\tau) > \lambda(T_2)$.

***P1.3*** $T^* = \max(T_1, T_2)$ *when* $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ *and* $\lambda(\tau) > \lambda(T_2)$.

***P1.4*** $T^* = 0$ *when* $\lambda(0) < \lambda(T_1) \leq \lambda(\tau)$ *and* $\lambda(\tau) < \lambda(T_2)$.

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

### 4.2. Optimal software release policies: cost and reliability requirement

Now we derive the optimal release time polices based on the minimization of the software cost with respect to the desired level of reliability. Consider the relations $R(x/T_1) = R_0$, where $T_{R_1}$ is the optimal release time of the software with respect to $T_1$, and $R(x/T_2) = R_0$, where $T_{R_2}$ is the optimum release time with respect to $T_2$. Using these relations, $R(x/T_1) = R_0$ and $R(x/T_2) = R_0$, in Eq. (6), we can get $T_{R_1}$ and $T_{R_2}$ as:

$$T_{R_1} = \frac{1}{b_1}\left\{\ln(m_1(x)) - \ln\ln\left(\frac{1}{R(x/T_1) = R_0}\right)\right\} \tag{15}$$

and

$$T_{R_2} = \frac{1}{b_1}\left\{\ln(m_2(x)) + (b_1 - b_2)\tau - \ln\ln\left(\frac{1}{R(x/T_2) = R_0}\right)\right\}. \tag{16}$$

After getting $T_{R_1}$ and $T_{R_2}$, we can now derive the following optimal release policies by considering both the minimization of the total expected software cost and the desired level of the software reliability.

**Optimum release policy 2**

***P2.1 If*** $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ ***and*** $\lambda(\tau) < \lambda(T_2)$ ***and***
$R(x/0) > R(x/T_{R_1}) = R_0$ ***then*** $T^* = T_1$.

***P2.2 If*** $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ ***and*** $\lambda(\tau) < \lambda(T_2)$ ***and***
$R(x/0) < R(x/T_{R_1}) = R_0$ ***then*** $T^* = \max\{T_1, T_{R_1}\}$.

***P2.3 If*** $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ ***and*** $\lambda(\tau) > \lambda(T_2)$ ***and***
$R(x/0) < R(x/T_{R_1}) = R_0$ ***then*** $T^* = \max\{T_1, T_2, T_{R_1}\}$.

***P2.4 If*** $\lambda(0) < \lambda(T_1) \leq \lambda(\tau)$ ***and*** $\lambda(\tau) > \lambda(T_2)$ ***and***
$R(x/0) > R(x/T_{R_2}) = R_0$ ***then*** $T^* = T_2$.

***P2.5 If*** $\lambda(0) < \lambda(T_1) \leq \lambda(\tau)$ ***and*** $\lambda(\tau) > \lambda(T_2)$ ***and***
$R(x/0) > R(x/T_{R_2}) = R_0$ ***then*** $T^* = \max\{T_2, T_{R_2}\}$.

***P2.6 If*** $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ ***and*** $\lambda(\tau) > \lambda(T_2)$ ***and***
$R(x/0) > R(x/T_{R_2}) = R_0$ ***then*** $T^* = \max\{T_1, T_2\}$.

***P2.7 If*** $\lambda(0) > \lambda(T_1) \geq \lambda(\tau)$ ***and*** $\lambda(\tau) < \lambda(T_2)$ ***and***
$R(x/0) > R(x/T_{R_2}) = R_0$ ***then*** $T^* = 0$.

## 5. Numerical illustration

With the help of the J3 dataset given in [23] (see Table 1), we have estimated the parameters of the NHPP-CP model using the least square method as $\tau = 15, \hat{a} = 356.937$, $\hat{b}_1 = 0.071987$, and $\hat{b}_2 = 0.232653$. For the J3 dataset, the NHPP-CP model gives the best fit (see Figure 2). From the above estimated $\hat{b}_1$ and $\hat{b}_2$, it is clear that they are significantly different. We also observed that after week 15 of the testing phase, there was a change in the test procedure.

**Table 1.** The J3 dataset.

| Time (in weeks) (in weeks) | Number of failures | Cumulative no. of errors | Time intervals (in weeks) | Number of failures | Cumulative no. of errors |
|---|---|---|---|---|---|
| 1 | 4 | 4 | 21 | 13 | 272 |
| 2 | 12 | 16 | 22 | 19 | 291 |
| 3 | 15 | 31 | 23 | 10 | 301 |
| 4 | 9 | 40 | 24 | 5 | 306 |
| 5 | 28 | 68 | 25 | 5 | 311 |
| 6 | 29 | 97 | 26 | 5 | 316 |
| 7 | 8 | 105 | 27 | 7 | 323 |
| 8 | 7 | 112 | 28 | 7 | 330 |
| 9 | 4 | 116 | 29 | 1 | 331 |
| 10 | 8 | 124 | 30 | 3 | 334 |
| 11 | 9 | 133 | 31 | 1 | 335 |
| 12 | 12 | 145 | 32 | 2 | 337 |
| 13 | 8 | 153 | 33 | 0 | 337 |
| 14 | 4 | 157 | 34 | 2 | 339 |
| 15 | 14 | 171 | 35 | 9 | 348 |
| 16 | 19 | 190 | 36 | 1 | 349 |
| 17 | 23 | 213 | 37 | 0 | 349 |
| 18 | 12 | 225 | 38 | 0 | 349 |
| 19 | 22 | 247 | 39 | 0 | 349 |
| 20 | 12 | 259 | 40 | 1 | 350 |
| | | | 41 | 1 | 351 |

Figure 2 shows the goodness fit of the observed and estimated cumulative number of software errors. This gives a good fit and as per our discussion, the change almost agrees with that of [23]. Thus, we may conclude from Figure 2 that there is a change in the test procedure after week 14 by the plot of the observed failure rate.

Let us consider the following parameter values of the developed cost model. To discuss the impact of the coefficients on the total expected development cost, let us assume the parameter values for the cost model as follows:

$$c_0 = 100, c_t = 25, c_w = 5, \hat{a} = 356.937, \hat{b}_1 = 0.041987, \hat{b}_2 = 0.132653, \text{ and } \alpha = 0.01.$$

To study the various effects on the total expected development cost based on the coefficient, consider the parameter value in Eq. (12). From Table 2, a different warranty period with respect to the different testing time gives the respective expected total cost. Moreover, Table 2 shows that the expected total costs for different warranty periods increase with respect to the increasing software testing time. The pictorial representation of the expected total cost for different warranty periods for different periods of testing times is shown in Figure 3,

and it is noted that the optimal warranty period for the J3 dataset with respect to the assumed cost value is $T_w = 10$. Based on the different values of unit testing cost $c_t$ with respect to warranty period $T_w$, we obtained the optimal release times. Using the above parameter values, the optimal release times for optimal release policy 1 and optimal release policy 2 are shown in Tables 3 and 4, respectively.

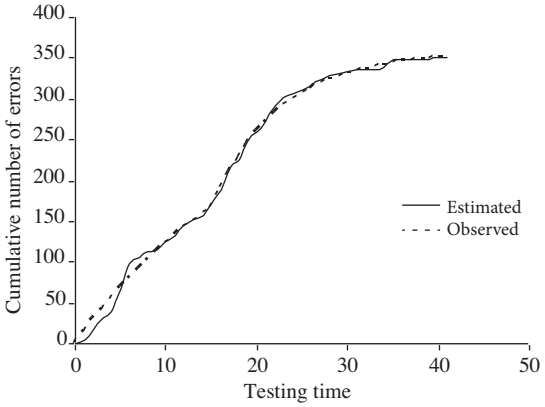


**Figure 2.** The observed and estimated cumulative number of software errors for the J3 dataset.

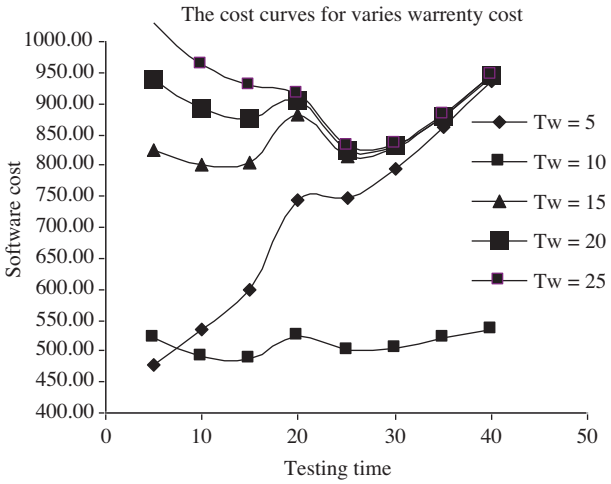


**Figure 3.** Software cost for different different periods of $T_w$.

**Table 2.** The total expected cost for periods of warranty $T_w$.

| T / $T_W$ | **5** | **10** | **15** | **20** | **25** |
|---|---|---|---|---|---|
| **5** | 476.34 | 519.90 | 823.79 | 940.43 | 1030.38 |
| **10** | 534.08 | 492.01 | 802.00 | 891.95 | 961.31 |
| **15** | 599.50 | 487.10 | 806.09 | 875.45 | 928.93 |
| **20** | 743.32 | 525.85 | 882.17 | 904.55 | 915.51 |
| **25** | 746.18 | 500.80 | 814.22 | 825.19 | 830.56 |
| **30** | 793.62 | 506.11 | 826.96 | 832.33 | 834.97 |
| **35** | 860.66 | 520.20 | 877.00 | 879.63 | 880.92 |
| **40** | 935.17 | 535.18 | 943.17 | 944.46 | 945.10 |

$$(c_0 = \$100, c_t = \$25, c_w = 5\ , \hat{a} = 356.937, \hat{b}_1 = 0.041987, \hat{b}_2 = 0.132653, \text{ and } \alpha = 0.01)$$

Tables 3 and 4 show that the optimal release time decreases for an increasing unit testing cost $c_t$ and warranty time $T_w$. This indicates that for a longer warranty period, a longer testing time is needed. Using Eqs. (15) and (16), it is easy to calculate the optimal release times $T_{R_1}$ and $T_{R_2}$, respectively, for the various values of the desired level of reliability $R_0$ and the operational period $x$. The different values of software reliability requirement $R_0$ and operational period $x$ provide optimal release time $(T_R)$, satisfying Eqs. (10) and (16). These optimal release times are shown in Table 5 and indicate that the optimal release time increases with a longer operational period. Figure 4 shows the different software reliability curves for the J3 dataset for different warranty periods.

**Table 3.** Optimal release policy 1.

| $c_t$ / $T_w$ | **5** | **10** | **15** | **20** | **25** |
|---|---|---|---|---|---|
| **5** | 34.26 | 29.03 | 25.97 | 0.00 | 0.00 |
| **10** | 42.97 | 32.04 | 28.98 | 26.81 | 25.13 |
| **15** | 49.87 | 33.36 | 30.11 | 27.94 | 26.26 |
| **20** | 54.09 | 37.58 | 30.60 | 28.43 | 26.75 |
| **25** | 56.90 | 40.39 | 30.83 | 28.67 | 26.98 |

**Table 4.** Optimal release policy 2.

| $c_t$ / $T_w$ | **5** | **10** | **15** | **20** | **25** |
|---|---|---|---|---|---|
| **5** | 34.26 | 29.03 | 25.97 | 24.66 | 24.66 |
| **10** | 42.97 | 32.04 | 28.98 | 26.81 | 25.13 |
| **15** | 49.87 | 33.36 | 30.11 | 27.94 | 26.26 |
| **20** | 54.09 | 37.58 | 30.60 | 28.43 | 26.75 |
| **25** | 56.90 | 40.39 | 30.83 | 28.67 | 26.98 |

($c_0 = \$100, c_t = \$25, \hat{a} = 356.937, \hat{b}_1 = 0.041987, \hat{b}_2 = 0.132653,$ and $\alpha = 0.01$)

For optimal release policy 2, the optimal release time decreases for different warranty periods. $c_0 = 100, c_w = 5, T_w = 15, c_t = 25, \hat{a} = 356.937, \hat{b}_1 = 0.041987, \hat{b}_2 = 0.132653, R_0 = 0.9$, and $\alpha = 0.01$ (see Tables 4 and 5).

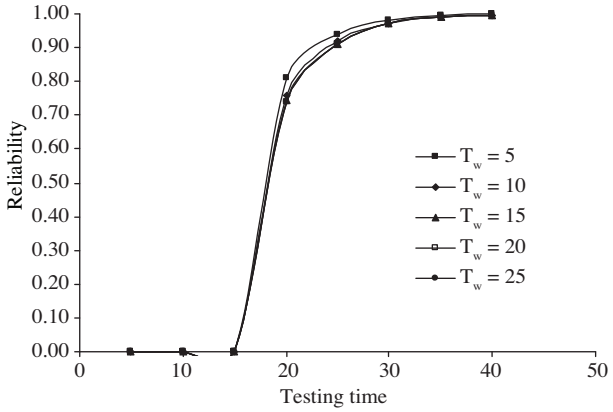


**Figure 4.** Reliability curve for different warranty periods.

**Table 5.** Optimal release time ($T_R$) for varying values of $R_0$ and $x$.

| $R_0$ / x | **0.3** | **0.4** | **0.5** | **0.6** | **0.7** | **0.8** | **0.9** |
|---|---|---|---|---|---|---|---|
| **5** | 12.50 | 13.67 | 14.87 | 16.18 | 17.73 | 19.74 | 22.97 |
| **10** | 13.67 | 14.84 | 16.04 | 17.35 | 18.89 | 20.91 | 24.14 |
| **15** | 13.97 | 15.15 | 16.35 | 17.66 | 19.20 | 21.22 | 24.44 |
| **20** | 14.07 | 15.24 | 16.44 | 17.75 | 19.29 | 21.31 | 24.54 |
| **25** | 14.09 | 15.27 | 16.47 | 17.78 | 19.32 | 21.34 | 24.56 |

($\hat{a} = 356.937, \hat{b}_1 = 0.041987, \hat{b}_2 = 0.132653, \tau = 15$ and $\alpha = 0.01$)

Figure 5 provides an example for optimal release policy 2 and shows the optimal release time with respect to reliability requirement $R_0$. In the case of $T_w = 15, c_t = 25$, we derive $T_1 = 26.26$ (see Table 1).

We have also obtained $T_{R_2} = 24.44$ (see Table 4) for the values of $R_0 = 0.9$ and $x = 15$. Thus, we have:

$$\lambda(0) = 15, \lambda(\tau) = \lambda(10) = 7, T_1 = 11.54, T_2 = 26.26, \lambda(T_1) = \lambda(11.54) = 9,$$

$$\lambda(T_2) = \lambda(26.26) = 6.$$

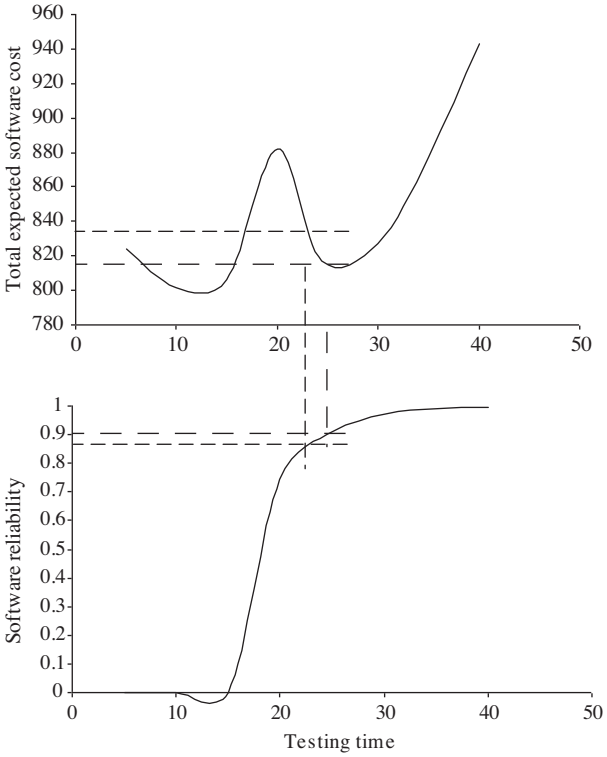


**Figure 5.** Optimal release time problem.

Hence, the obtained values satisfy P2.5 of optimal release policy 2, since $\lambda(0) = 15¿9{=}\lambda(T_1) > 7 = \lambda(10) = \lambda(\tau)$ and $\lambda(\tau) = \lambda(10) = 7 > 6 = \lambda(T_2) = \lambda(26.26)$

and $R(x = 15/0) = 0 < 0.9 = R_0 = R(x = 15/T_{R_2}) = R(x = 15/24.44)$.

Thus, the optimal release time for the J3 dataset for warranty period $T_w = 15$ is

$T^* = \max\{T_1, T_2, T_R\} = \max\{11.54, 26.26, 24.44\} = T_2 = 26.26$ (see Figure 5).

## 6. Conclusion

In this paper, we have developed a new cost model for software reliability with respect to the change-point and warranty cost phenomenon. Using this cost model, we have derived 2 optimal release time policies. Optimal release policy 1 was formulated by minimizing the total expected software cost with respect to the unit testing cost, discount rate of the total software cost, and maintenance cost during the warranty period. Optimal release policy 2 was formulated by minimizing the total expected software cost subject to the desired level of software reliability, with respect to unit testing cost, discount rate of the total software cost, and maintenance cost during the warranty period. The sensitivity analyses of the optimal release policies were also discussed for the real software J3 dataset. The numerical example presented in Section 5 showed that the optimal release time decreases for increasing warranty periods. Using optimal release policy 2, for the values

$c_0 = 100, c_w = 5, T_w = 15, c_t = 25, \hat{a} = 356.937, \hat{b}_1 = 0.041987, \hat{b}_2 = 0.132653, R_0 = 0.9,$ and $\alpha = 0.01$, the optimal release time of the software is given as 26.26 weeks. This result is perfectly matched with the graphical representation of optimal release policy 2 in Figure 5. Thus, the numerical example supports the derived optimal release policies. Predicting an accurate optimal release time for the software system with respect to the change-point, warranty period, and warranty cost is a new technique. This newly developed cost

model will be suitable for any software manager when choosing the best software economic policy based on cost, software reliability, warranty period, and change-point. In the future, we will develop a cost model for imperfect debugging software reliability in regard to the change-point and warranty cost phenomenon.

**Acknowledgment**

The author is extremely grateful to the learned editor and referees for their constructive and insightful suggestions, which was of great help for improving the standards of this paper.

## References

[1] Y.K. Malaiya, P.K. Srimani, Software Reliability Models: Theoretical Developments, Evaluation and Applications, Los Alamitos, CA, USA, IEEE Computer Society Press, 1990.

[2] J.D. Musa, A. Iannino, K. Okumoto, Software Reliability: Measurement, Prediction, and Application, New York, McGraw-Hill, 1987.

[3] H. Pham, "A software cost model with imperfect debugging random life cycle and penalty cost", International Journal of Systems Science, Vol. 27, pp. 455–463, 2003.

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

[5] D.R. Prince Williams, "Study of the warranty cost model for software reliability with an imperfect debugging phenomenon", Turkish Journal of Electrical Engineering & Computer Sciences, Vol. 15, pp. 1048–1052, 2007.

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

[7] O. Tal, C. McCollin, A. Bendell, "An optimal statistical testing policy for software reliability demonstration of safety critical systems", European Journal of Operational Research, Vol. 137, pp. 544–557, 2002.

[8] S. Yamada, "Optimal release problems with warranty period based on a software maintenance cost model", Transactions of Information Processing Society of Japan, Vol. 35, pp. 2197–2202, 1994.

[9] S. Yamada, M. Kimura, E. Terane, S. Osaki, "Optimal software release problems with life-cycle distribution and discount rate", Transactions of Information Processing Society of Japan, Vol. 34, pp. 1188–1197, 1993 (in Japanese).

[10] B.C. Cho, K.S. Park, "An optimal time for software testing under the user's requirement of failure-free demonstration before release", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E77-A, pp. 563–570, 1994.

[11] K. Okumoto, A.L. Goel, "Optimum release time for software system based on reliability and cost criteria", Journal of Systems and Software, Vol. 14, pp. 315–318, 1980.

[12] S. Yamada, S. Osaki, "Optimal software release policies with simultaneous cost and reliability requirements", European Journal of Operational Research, Vol. 31, pp. 46–51, 1987.

[13] P.K. Kapur, S. Agarwal, R.B. Garg, "Bicriterion release policy for exponential software reliability growth model", Proceedings of the 3rd International Symposium on Software Reliability Engineering, Vol. 28, pp. 165–180, 1994.

[14] M. Jain, B.R. Handa, Cost analysis for repairable units under hybrid warranty. In: M.L. Agarwal, K. Sen, eds., Recent Developments in Operational Research, New Delhi, Narosa Publishing House, pp. 149–165, 2001.

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

[16] A.L. Goel, K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures", IEEE Transactions on Reliability, Vol. R-28, pp. 206–211, 1979.

[17] S. Yamada, M. Ohba, S. Osaki, "S-shaped reliability growth modeling for software error detection", IEEE Transactions on Reliability, Vol. 12, pp. 475–484, 1983.

[18] Y.P. Chang, "Estimation of parameters for nonhomogeneous Poisson process: software reliability with change-point", Communications in Statistics - Simulation and Computation, Vol. 30, pp. 623–635, 2006.

[19] 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.

[20] H. Pham, Software Reliability and Testing, Los Alamitos, CA, USA, IEEE Computer Society Press, 1990.

[21] M.L. Shooman, Software Engineering: Design, Reliability, and Management, New York, McGraw-Hill, 1983.

[22] M. Xie, Software Reliability Modeling, Singapore, World Scientific, 1991.

[23] M.R. Lyu, Handbook of Software Reliability Engineering, New York, IEEE Computer Society Press and McGraw-Hill, 1996.