Discrete Optimal Testing/Maintenance Policy in a Software Development Project

林坂弘一郎，土肥正
Koichiro Rinsaka and Tadashi Dohi
Department of Information Engineering, Graduate School of Engineering, Hiroshima University, Japan

1 Introduction

It is important to determine the optimal time when software testing should be stopped and when the system should be delivered to a user or a market. This problem, called \textit{optimal software release problem}, plays a central role for the success or failure of a software development project. Okumoto and Goel [1] assumed that the number of software faults detected in the testing phase is described by an exponential software reliability model based on a non-homogeneous Poisson process (NHPP) [2], and derived an optimal software release time which minimizes the total expected software cost. Koch and Kubat [3] considered the similar problem for the other software reliability model by Jelinski and Moranda [4]. Bai and Yun [5] calculated the optimal number of faults detected before the release under the Jelinski and Moranda model. Many authors formulated the optimal software release problems based on different model assumptions and/or several software reliability models [6, 7].

It is difficult to detect and remove all faults remaining in a software during the actual testing phase, because exhaustive testing of all executable paths in a general program is impossible. Once the software is released to users, however, the software failures may occur even in the operational phase. It is common for software developers to provide maintenance service during the period when they are still responsible for fixing software faults causing failures. In order to carry out the maintenance in the operational phase, the software developer has to keep a software maintenance team. At the same time, the management cost in the operational phase should be reduced as much as possible, but at the same time the human resources should be utilized effectively. Although the problem which determines the maintenance period is important from the practical point of view, only a very few authors paid their attention to this problem.

Kimura et al. [8] considered the optimal software release problem in the case where the software warranty period is a random variable. Pham and Zhang [9] developed a software cost model with both warranty and risk. They focused on the problem of determining when to stop the software testing under a warranty contract. However, it is noted that the software developer has to design the warranty contract itself and often provides the posterior service for users after software failures. Dohi et al. [10] formulated the problem for determining the optimal software warranty period which minimizes the total expected software cost under the assumption that the debugging process in the testing phase is described by an NHPP. Sandoh and Rinsaka [11] considered the design problem of a maintenance service contract by regarding as the Stackelberg game between a software agent and a software user. Since the user's operational environment is not always same as that assumed in the software development phase, however, the above literature did not take account of the difference between two different phases.

Several reliability assessment methods during the operational phase have been proposed by some authors [12, 13]. Rinsaka and Dohi [14] developed a continuous time model for designing the optimal testing and maintenance periods, where the difference between the software testing environment and the operational environment are reflected.

In this paper, we focus on the optimal software testing/maintenance problem considered in Rinsaka and Dohi [14], and develop a stochastic model in discrete circumstance, where the difference between the software testing environment and the operational environment can be characterized by an environment factor (see Okamura et al. [13]). More precisely, the total expected software cost is formulated via the discrete NHPP type of software reliability models [16, 17, 18]. In the special case with the geometric fault-detection time distribution, we derive analytically the optimal testing period (release time) which minimizes the total expected software cost under a milder condition. We call the time length to complete the operational maintenance after the release a \textit{planned maintenance limit}, and also derive the optimal planned maintenance limit which minimizes the total expected software cost. In numerical examples with real data, we calculate numerically the joint optimal testing/maintenance policy, combined by testing period and planned maintenance limit.
2 Model Description

First, we make the following assumptions on the software fault-detection process:

(a) In each time when a software failure occurs, the software fault causing the failure can be detected and removed immediately.

(b) The number of initial faults contained in the software program, $N_0$, is given by the Poisson distributed random variable with mean $\omega (>0)$.

(c) The time to detect each software fault is independent and identically distributed nonnegative discrete random variable with probability mass function $p_i (i=0,1,2,\cdots)$ and probability distribution function $P(i) = \sum_{k=1}^{i} p_k$, where $0 \leq p_i \leq 1$ and $0 \leq P(i) \leq 1$.

Let $\{N_i, i=1,2,\cdots\}$ be the cumulative number of software faults detected up to time $i$. From the above assumptions, the probability mass function of $N_i$ is given by

$$\Pr\{N_i = m\} = \frac{\{\omega \sum_{k=1}^{i} p_k\}^m e^{-\omega \sum_{k=1}^{i} p_k}}{m!}$$

Hence, the stochastic process $\{N_i, i=1,2,\cdots\}$ is equivalent to a discrete time NHPP with mean value function $\omega P(i)$.

Suppose that a software testing is started at time $i = 0$ and terminated at time $i = n_0$ ($\geq 0$). After completing the software testing, the software product is released to a user or the market. The time length of software life cycle $n_L$ ($>0$) is a known constant in advance and is assumed to be sufficiently larger than $n_0$. More precisely, the software life cycle is measured from the point of time $n_0$. The software developer is responsible to the maintenance service for all the software failures that may occur during the software life cycle under a maintenance contract. We suppose that the project manager decides to break up the maintenance team at time $n_0 + n_W$ for reduction of the operational cost to keep it, but a large amount of debugging cost during $(n_0 + n_W, n_0 + n_L]$ may be needed if the software failure occurs. Let $n_W$ be the planned maintenance limit denoting the time length to complete the operational maintenance after the release a planned maintenance limit. Further, we define the following cost components:

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

$c_W$ ($>0$): cost to remove each fault before the planned maintenance limit,

$c_L$ ($>0$): cost to remove each fault after the planned maintenance limit,

$k_0$ ($>0$): testing cost per unit of time,

$k_W$ ($>0$): operational cost to keep the maintenance team per unit of time.

In the following section, we formulate the total expected software cost by introducing the above cost factors in testing and operational phases. We derive the discrete optimal software testing period $n_0^*$ or the discrete optimal planned maintenance limit $n_W^*$ which minimizes the total expected software cost at the end of the software life cycle. Then, we calculate the joint optimal policy $(n_0^{**}, n_W^{**})$ combined by both testing period and planned maintenance limit.

3 Total Expected Software Cost

We formulate the total expected software cost which can occur in both testing and operational phases. In the operational phase, we consider two cost factors; the maintenance cost due to the software failure and the operational cost to keep the maintenance team.

From Eq.(1), the probability mass function of the number of software faults detected during the testing phase is given by

$$\Pr\{N_{n_0} = m\} = \frac{\{\omega P(n_0)\}^m}{m!} e^{-\omega P(n_0)}.$$ (2)
It should be noted that the operational environment after the release may differ from the debugging environment in the testing phase. This difference is similar to that between the accelerated life testing environment and the normal operating environment for hardware products. We introduce the environment factor \( a ( > 0 ) \) which represents the relative severity in the operational environment after the release, and assume that the times in the testing phase and the operational phase have a proportional relationship. Namely, the time length \( n \) in the operational phase corresponds to \( a \times n \) in the testing phase. Under the above assumption, \( a = 1 \) means the equivalence between the testing and operational environments. On the other hand, \( a > 1 \) (\( a < 1 \)) implies that the operational environment is severer (looser) than the testing environment. Okamura et al. [13] apply this technique to model the operational phase of the software, and estimate the software reliability through an example in the actual software development project. The probability mass function of the number of software faults detected before the planned maintenance limit is given by

\[
\Pr\{N_{n_{0}+n_{W}} - N_{n_{0}} = m\} = \frac{\{\omega \{P(n_{0} + [an_{W}]) - P(n_{0})\}\}^{m}}{m!} e^{-\omega \{P(n_{0} + [an_{W}]) - P(n_{0})\}}, \tag{3}
\]

where, \([ \cdot ]\) is the Gaussian integer.

Similarly, the fault-detection process of the software after the planned maintenance limit is expressed by

\[
\Pr\{N_{n_{0}+n_{L}} - N_{n_{0}+n_{W}} = m\} = \frac{\{\omega \{P(n_{0} + [an_{L}]) - P(n_{0} + [an_{W}])\}\}^{m}}{m!} e^{-\omega \{P(n_{0} + [an_{L}]) - P(n_{0} + [an_{W}])\}}. \tag{4}
\]

From Eqs.\((2)\), \((3)\) and \((4)\), the total expected software cost is given by

\[
C(n_{0}, n_{W}) = k_{0}n_{0} + c_{0} \omega \{1 - (1 - b)^{n_{0}}\} + k_{W}n_{W} + c_{W} \omega \{P(n_{0} + [an_{W}]) - P(n_{0})\} + c_{L} \omega \{P(n_{0} + [an_{L}]) - P(n_{0} + [an_{W}])\}. \tag{5}
\]

### 4 Determination of the Optimal Policies

In this section we derive the optimal testing period \( n_{0}^{*} \) or the optimal planned maintenance limit \( n_{W}^{*} \) which minimizes the total expected software cost incurred to the software developer at the end of software life cycle. Suppose that the time to detect each software fault obeys the geometric distribution

\[
p_{i} = b(1 - b)^{i-1} \tag{6}
\]

with parameter \( b \) (\( 0 < b < 1 \)). In this case, the total expected software cost in Eq.\((5)\) becomes

\[
C(n_{0}, n_{W}) = k_{0}n_{0} + c_{0} \omega \{(1 - (1 - b)^{n_{0}})\} + k_{W}n_{W} + c_{W} \omega \{(1 - b)^{n_{0}} - (1 - b)^{n_{0} + [an_{W}]}\} + c_{L} \omega \{(1 - b)^{n_{0} + [an_{W}]} - (1 - b)^{n_{0} + [an_{L}]}\}. \tag{7}
\]

We make the following assumptions:

\textbf{(A-I)} \( a \) is positive and an integer value,

\textbf{(A-II)} \( c_{L} > c_{W} > c_{0} \),

\textbf{(A-III)} \( c_{W} \{1 - (1 - b)^{an_{L}}\} > c_{0} \),

\textbf{(A-IV)} \( c_{W} \{1 - (1 - b)^{an_{W}}\} + c_{L} \{(1 - b)^{an_{W}} - (1 - b)^{an_{L}}\} > c_{0} \).

Define

\[
Q(n_{W}) = k_{0} + \omega b \{c_{0} - c_{W} (1 - (1 - b)^{an_{W}}) - c_{L} ((1 - b)^{an_{W}} - (1 - b)^{an_{L}})\}. \tag{8}
\]

Then the following result provides the optimal software testing policy which minimizes the total expected software cost.
Theorem 1: When the software fault-detection time distribution follows the geometric distribution with parameter \( b \) (0 < \( b < 1 \)), under the assumptions (A-I) to (A-IV), the optimal software testing period (release time) which minimizes the total expected software cost is given as follows:

1. If \( Q(n_{W}) < 0 \), then there exist \( \) (at least one, at most two) finite optimal software testing periods (release times) \( n_{o}^{*} \) such that \( \).

2. If \( Q(n_{W}) \geq 0 \), then the optimal policy is \( n_{l}^{*} = 0 \) with

\[
C(n_{o}^{*}, n_{W}) = k_{W}n_{W} + c_{W}\omega \{1 - (1 - b)^{n_{W}}\} + c_{L}\omega \{(1 - b)^{n_{W}} - (1 - b)^{n_{L}}\}.
\]

Furthermore, the following result provides the optimal planned maintenance limit which minimizes the total expected software cost.

Theorem 2: When the software fault-detection time distribution follows the geometric distribution with parameter \( b \) (0 < \( b < 1 \)), under the assumptions (A-I) and (A-II), the optimal planned maintenance limit which minimizes the total expected software cost is given as follows:

1. If \( k_{W} \geq (c_{L} - c_{W})\omega \{1 - (1 - b)^{a}\} \) \( (1 - b)^{n_{o}^{*}} \), then the optimal policy is \( n_{l}^{*} = 0 \) with

\[
C(n_{o}^{*}, n_{W}) = k_{o}n_{o} + c_{o}\omega \{1 - (1 - b)^{n_{o}^{*}}\} + c_{L}\omega \{(1 - b)^{n_{o}^{*}} - (1 - b)^{n_{o}^{*} + an_{L}}\}\.
\]

2. If \( k_{W} < (c_{L} - c_{W})\omega \{1 - (1 - b)^{a}\} \) \( (1 - b)^{n_{o}^{*}} \) \( (1 - b)^{n_{o}^{*} + an_{L}} \) \( \omega \{(1 - b)^{n_{0}^{*}} - (1 - b)^{n_{0}^{*} + an_{L}}\} \), then there exist (at least one, at most two) optimal planned maintenance limits \( n_{l}^{*} \) \( (0 < n_{l}^{*} < n_{L}) \) which minimizes the total expected software cost.

3. If \( k_{W} \leq (c_{L} - c_{W})\omega \{1 - (1 - b)^{a}\} \) \( (1 - b)^{n_{o}^{*} + an_{L}} \), then we have \( n_{l}^{*} = n_{L} \) with

\[
C(n_{o}^{*}, n_{W}) = k_{o}n_{o} + c_{o}\omega \{1 - (1 - b)^{n_{o}^{*}}\} + k_{W}n_{L} + c_{W}\omega \{(1 - b)^{n_{o}^{*}} - (1 - b)^{n_{o}^{*} + an_{L}}\}.
\]

5 Numerical Examples

Based on 351 software fault (41 week) data observed in the real software testing process [19], we calculate numerically the optimal testing period \( n_{o}^{*} \) and the optimal planned maintenance limit \( n_{l}^{*} \) which minimize the total expected software cost. Further, we compute the joint optimal policy \( (n_{o}^{*}, n_{l}^{*}) \) minimizing \( C(n_{o}, n_{W}) \). For the software fault-detection time distribution, we apply three probability distributions; geometric, negative binomial and discrete Weibull distributions. The probability mass functions for negative binomial and discrete Weibull are given by

\[
p_{i} = \left( \frac{h + i - 2}{h - 1} \right) b^{i} (1 - b)^{h-i-1},
\]

and

\[
p_{i} = b^{h} - b^{h+i},
\]

respectively, where \( h \geq 0 \). For negative binomial and discrete Weibull distributions, we consider the case of \( h = 2 \).

Suppose that the unknown parameters in the software reliability models are estimated by the method of maximum likelihood. Then, we have the estimates \( (\hat{\omega}, \hat{b}) = (413.305, 0.0451012) \) for the geometric model, \( (\hat{\omega}, \hat{b}) = (364.234, 0.116255) \) for the negative binomial model and \( (\hat{\omega}, \hat{b}) = (351.871, 0.996436) \) for the discrete Weibull model. Figure 1 shows the actual software fault data and the behavior of estimated mean value functions. Since the environment factor \( a \) is a subjective parameter which should be estimated from the past development trend, we assume that the value of \( a \) is known. For the other model parameters, we assume: \( k_{0} = 2.0, k_{W} = 1.0, c_{0} = 5.0, c_{W} = 10.0, c_{L} = 50.0 \) and \( n_{L} = 200 \).

Table 1 presents the dependence of environment factor \( a \) on the optimal testing period \( n_{o}^{*} \) when \( n_{W} = 20 \). As the environment factor monotonically increases, \( i.e., \) the operational circumstance tends to be severe, it is observed that the optimal testing period \( n_{o}^{*} \) and its associated minimum total expected software cost \( C(n_{o}^{*}, 20) \) decrease for both geometric and negative binomial models. For the discrete Weibull model, since it is estimated that the software fault is hardly discovered after the release as shown.
Figure 1: Behavior of actual software fault data and estimated mean value functions.

Table 1: Optimal software testing period for varying environment factor.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$n^*_D$</th>
<th>$C(n^*_D, 20)$</th>
<th>$n^*_D$</th>
<th>$C(n^*_D, 20)$</th>
<th>$n^*_D$</th>
<th>$C(n^*_D, 20)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>122</td>
<td>2375</td>
<td>65</td>
<td>1990</td>
<td>41</td>
<td>1867</td>
</tr>
<tr>
<td>0.75</td>
<td>119</td>
<td>2367</td>
<td>62</td>
<td>1983</td>
<td>40</td>
<td>1865</td>
</tr>
<tr>
<td>1.00</td>
<td>115</td>
<td>2359</td>
<td>59</td>
<td>1977</td>
<td>39</td>
<td>1865</td>
</tr>
<tr>
<td>1.25</td>
<td>111</td>
<td>2352</td>
<td>57</td>
<td>1973</td>
<td>39</td>
<td>1865</td>
</tr>
<tr>
<td>1.50</td>
<td>108</td>
<td>2345</td>
<td>56</td>
<td>1971</td>
<td>39</td>
<td>1865</td>
</tr>
<tr>
<td>2.00</td>
<td>101</td>
<td>2333</td>
<td>54</td>
<td>1967</td>
<td>39</td>
<td>1865</td>
</tr>
<tr>
<td>3.00</td>
<td>93</td>
<td>2315</td>
<td>53</td>
<td>1966</td>
<td>39</td>
<td>1865</td>
</tr>
</tbody>
</table>

Table 2: Optimal planned maintenance limit for varying environment factor.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$n^*_W$</th>
<th>$C(41, n^*_W)$</th>
<th>$n^*_W$</th>
<th>$C(41, n^*_W)$</th>
<th>$n^*_W$</th>
<th>$C(41, n^*_W)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>176</td>
<td>2648</td>
<td>62</td>
<td>2050</td>
<td>12</td>
<td>1863</td>
</tr>
<tr>
<td>0.75</td>
<td>128</td>
<td>2615</td>
<td>48</td>
<td>2028</td>
<td>8</td>
<td>1859</td>
</tr>
<tr>
<td>1.00</td>
<td>103</td>
<td>2584</td>
<td>38</td>
<td>2016</td>
<td>8</td>
<td>1856</td>
</tr>
<tr>
<td>1.25</td>
<td>88</td>
<td>2564</td>
<td>32</td>
<td>2008</td>
<td>8</td>
<td>1855</td>
</tr>
<tr>
<td>1.50</td>
<td>74</td>
<td>2549</td>
<td>28</td>
<td>2003</td>
<td>6</td>
<td>1854</td>
</tr>
<tr>
<td>2.00</td>
<td>59</td>
<td>2530</td>
<td>22</td>
<td>1996</td>
<td>5</td>
<td>1852</td>
</tr>
<tr>
<td>3.00</td>
<td>42</td>
<td>2509</td>
<td>16</td>
<td>1988</td>
<td>4</td>
<td>1850</td>
</tr>
</tbody>
</table>
Remarks of Concluding joint of team optimal of the software execution expected environment continuous-time-based reduced the developer environment monotonically maintenance operational planned the maintenance stage optimal tendency corresponding maintenance operational stage optimal environment dependence testing maintenance which which reliability proposed expected software period environment maintenance optimal detection when maintenance environment the keep software length cost-benefit planned total maintenance carry throughout software discrete maintenance service which same discussed formulated optimal factor have and the case Fig. 1, period all software controlled software expected maintenance small. software the life limit. Hence, we incurred the software testing the residual faults in software are detected and removed at the early stage in the operational phase as the operational environment becomes more severe. Then, the possibility that the software failure occurs in the latter stage of the operational phase may become small. Hence, the implication in which the software developer keeps the maintenance team becomes smaller toward the end of the life cycle.

Table 3 examines the dependence of environment factor $a$ on the joint optimal policy $(n_0^{**}, n_W^{**})$ combined by testing period and planned maintenance limit. Figure 2 illustrates the behavior of the expected cost for the geometric model when $a = 2.00$. It is observed from Table 3 that the optimal testing period $n_0^{**}$ decreases as the environment factor monotonically increases, but, the monotonicity of the optimal planned maintenance limit $n_W^{**}$ is not observed. It is also seen that the minimum total expected software cost $C(n_0^{**}, n_W^{**})$ decreases as the environment factor monotonically increases.

Tables 4, 5 and 6 present the dependence of the software reliability model parameter $b$ on the joint optimal policy $(n_0^{**}, n_W^{**})$. It is observed that the optimal testing time, optimal planned maintenance limit and its associated minimum total expected software cost decrease as the fault detection becomes easier.

### 6 Concluding Remarks

In this paper, we have assumed that the software developer was responsible to the maintenance service for all the software failures that occur during the software life cycle under the maintenance contract. In order to carry out the maintenance service in the operational phase, the software developer has to keep a software maintenance team. At the same time, the management cost in the operational phase has to be reduced as much as possible, but human resources should be utilized effectively. We have called the time length to complete the operational maintenance after the release the planned maintenance limit, and have controlled it in terms of cost-benefit analysis. We have developed the discrete model which represents the difference in the software execution environment during testing and operational phases, using the same method as the continuous-time-based reliability assessment modeling in the operational phase proposed by Okamura et al. [13]. Based on the discrete NHPP, we have formulated the total expected software cost incurred to the software developer at the end of software life cycle. The optimal testing period (release time) and optimal planned maintenance limit which minimize the total expected software cost have been derived. Then, throughout the numerical examples, we have discussed the joint optimal policy combined by testing period and planned maintenance limit.
Table 4: Optimal testing period/planned maintenance limit for geometric software reliability model with varying parameter $b$.

<table>
<thead>
<tr>
<th>$b$</th>
<th>$n_0^{**}$</th>
<th>$n_W^{**}$</th>
<th>$C(n_0^{<strong>}, n_W^{</strong>})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.042</td>
<td>95</td>
<td>37</td>
<td>2340</td>
</tr>
<tr>
<td>0.043</td>
<td>93</td>
<td>36</td>
<td>2335</td>
</tr>
<tr>
<td>0.044</td>
<td>92</td>
<td>35</td>
<td>2330</td>
</tr>
<tr>
<td>0.045</td>
<td>90</td>
<td>35</td>
<td>2325</td>
</tr>
<tr>
<td>0.046</td>
<td>88</td>
<td>34</td>
<td>2320</td>
</tr>
<tr>
<td>0.047</td>
<td>87</td>
<td>33</td>
<td>2315</td>
</tr>
<tr>
<td>0.048</td>
<td>86</td>
<td>32</td>
<td>2311</td>
</tr>
</tbody>
</table>

Table 5: Optimal testing period/planned maintenance limit for negative binomial software reliability model with varying parameter $b$.

<table>
<thead>
<tr>
<th>$b$</th>
<th>$n_0^{**}$</th>
<th>$n_W^{**}$</th>
<th>$C(n_0^{<strong>}, n_W^{</strong>})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.08</td>
<td>77</td>
<td>22</td>
<td>2025</td>
</tr>
<tr>
<td>0.09</td>
<td>70</td>
<td>19</td>
<td>2004</td>
</tr>
<tr>
<td>0.10</td>
<td>64</td>
<td>17</td>
<td>1987</td>
</tr>
<tr>
<td>0.11</td>
<td>59</td>
<td>15</td>
<td>1973</td>
</tr>
<tr>
<td>0.12</td>
<td>54</td>
<td>14</td>
<td>1961</td>
</tr>
<tr>
<td>0.13</td>
<td>50</td>
<td>13</td>
<td>1951</td>
</tr>
<tr>
<td>0.14</td>
<td>47</td>
<td>12</td>
<td>1942</td>
</tr>
</tbody>
</table>

Table 6: Optimal testing period/planned maintenance limit for discrete Weibull software reliability model with varying parameter $b$.

<table>
<thead>
<tr>
<th>$b$</th>
<th>$n_0^{**}$</th>
<th>$n_W^{**}$</th>
<th>$C(n_0^{<strong>}, n_W^{</strong>})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.999</td>
<td>72</td>
<td>10</td>
<td>1926</td>
</tr>
<tr>
<td>0.998</td>
<td>53</td>
<td>7</td>
<td>1880</td>
</tr>
<tr>
<td>0.997</td>
<td>44</td>
<td>6</td>
<td>1860</td>
</tr>
<tr>
<td>0.996</td>
<td>38</td>
<td>5</td>
<td>1847</td>
</tr>
<tr>
<td>0.995</td>
<td>35</td>
<td>4</td>
<td>1838</td>
</tr>
<tr>
<td>0.994</td>
<td>32</td>
<td>4</td>
<td>1832</td>
</tr>
<tr>
<td>0.993</td>
<td>30</td>
<td>4</td>
<td>1827</td>
</tr>
</tbody>
</table>
Figure 2: Behavior of the total expected software cost for geometric software reliability model $(a = 2.00)$. 

References


