Original citation:

Permanent WRAP url:
http://wrap.warwick.ac.uk/59044

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work of researchers of the University of Warwick available open access under the following conditions.

This article is made available under the Creative Commons Attribution 3.0 (CC BY 3.0) license and may be reused according to the conditions of the license. For more details see: http://creativecommons.org/licenses/by/3.0/

A note on versions:
The version presented in WRAP is the published version, or, version of record, and may be cited as it appears here.

For more information, please contact the WRAP Team at: publications@warwick.ac.uk
Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction

R. Peng a,*, Y.F. Li b, W.J. Zhang c, Q.P. Hu d

a Dongling School of Economics & Management, University of Science & Technology Beijing, China
b Warwick Business School, The University of Warwick, Coventry CV4 7AL, UK
c Ecole Centrale Paris-SUPELEC, Paris, France
d Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China

ARTICLE INFO

Article history:
Received 2 May 2012
Received in revised form 3 January 2014
Accepted 5 January 2014
Available online 11 January 2014

Keywords:
Software reliability
Fault detection process
Fault correction process
Testing effort function
Imperfect debugging

ABSTRACT

This paper studies the fault detection process (FDP) and fault correction process (FCP) with the incorporation of testing effort function and imperfect debugging. In order to ensure high reliability, it is essential for software to undergo a testing phase, during which faults can be detected and corrected by debuggers. The testing resource allocation during this phase, which is usually depicted by the testing effort function, considerably influences not only the fault detection rate but also the time to correct a detected fault. In addition, testing is usually far from perfect such that new faults may be introduced. In this paper, we first show how to incorporate testing effort function and fault introduction into FDP and then develop FCP as delayed FDP with a correction effort. Various specific paired FDP and FCP models are obtained based on different assumptions of fault introduction and correction effort. An illustrative example is presented. The optimal release policy under different criteria is also discussed.

© 2014 The Authors. Published by Elsevier Ltd. All rights reserved.

1. Introduction

As software is becoming more and more widely used, both the functionality and the correctness of software are of great concern. In order to ensure high reliability, testing is usually conducted, during which faults in software manifest by causing failures and can be detected and removed by debuggers [9,4,15]. On the other hand, it is almost impossible to make bug-free software even though scientific and disciplined development practices are followed. During the last 30 years, many software reliability growth models (SRGMs) have been proposed as a tool to track the reliability growth trend of the software testing process [16,3,35,23]. SRGMs are very useful in the sense that they can help management making critical decisions, such as testing resource allocation and the determination of software release time [19,13,12].

Testing consumes a large amount of resources, such as manpower and CPU hours, which are usually not constantly allocated during testing phase. The function that describes how testing resources are distributed is usually referred to as testing effort function (TEF) and it has been incorporated into software reliability studies by some researchers [30,16,17]. Yamada et al. [34] pointed out that the TEF could be described by a Weibull-type distribution, which actually includes Exponential curve, Rayleigh curve and Weibull curve. Weibull-type curve can well fit most data and is often used in the field of software reliability modeling [7]. Logistic TEF is used instead of Weibull-type TEF by some researchers and appeared to be fairly accurate in describing the consumption of testing effort [24,5,8].

Generally a detected fault cannot be corrected immediately and the time required to correct a detected fault is usually called debugging lag/delay. The idea of modeling the fault correction process (FCP) was first proposed in Schneidewind [28], in which it was modeled as a separate process following the fault detection process (FDP) with a constant time lag. Based on this framework, Xie et al. [33] and Wu et al. [31] proposed several paired FDP and FCP models through incorporating other variants of debugging delay. Later, Hwang and Pham [10] developed a generalized NHPP model considering quasi-renewal time-delay fault removal. Jia et al. [14] proposed a Markovian software reliability model considering the fault correction process. However, the influence of testing effort on debugging lag is not considered in these papers. Intuitively, the time needed to correct a detected fault, or the debugging lag, tends to be shorter if more testing effort is allocated during the period between detection and correction of the fault. Thus it is more reasonable to incorporate testing effort function into the modeling framework on both FDP and FCP.
Moreover, the debugging process is usually far from perfect and actually many faults encountered by customers are those introduced during debugging [36,29,6,26]. It is essential to incorporate imperfect debugging into FDP and FCP models [32,2,18].

In this paper, a framework is proposed to develop testing effort dependent FDP and FCP models with the consideration of imperfect debugging. The rest of this paper is organized as follows. In Section 2, a framework is proposed to obtain testing effort dependent paired FDP and FCP models with the consideration of fault introduction. In Section 3, several specific models are derived based on different assumptions of fault introduction and the correction effort. In Section 4, several commonly used testing effort functions are reviewed. In Section 5, an illustrative example is presented. The optimal release policy under different criteria is studied in Section 6. Conclusions and discussions are presented in Section 7.

2. Testing effort dependent FDP and FCP models with fault introduction

The expected total number of faults at time \( t \) is denoted by the fault content rate function \( a(t) \), which is the sum of the number of initial faults in the software \( a(0) \) and the number of faults introduced during time interval \( [0, t] \). We use \( w(t) \) to denote the cumulative testing effort consumed till time \( t \).

\[ m_d(t) = \frac{dmd(t)}{dt} = b(t)w(t)(a(t) - m_d(t)) \tag{1} \]

where \( b(t) \) is the current fault detection rate per unit of testing effort at time \( t \) and \( w(t) \) is the current testing effort expenditure. Following (1) gives

\[ m_d(t) = a(t) - a \exp \left( - \int_0^t b(x)w(x)dx \right) - \exp \left( - \int_0^t b(x)w(x)dx \right) \int_0^t a'(x) \exp \left( \int_0^x b(y)w(y)dy \right) dx \tag{2} \]

where \( a'(x) = da(x)/dx \). Various \( m_d(t) \) can be derived based on different assumptions of \( a(t), b(t) \) and \( w(t) \). \( \lambda_d(t) \) can be obtained by substituting (2) into the right hand side of (1) as

\[ \lambda_d(t) = \frac{dmd(t)}{dt} = ab(t)w(t) \exp \left\{ - \int_0^t b(x)w(x)dx \right\} \left( 1 + \int_0^t \frac{a'(x)}{a} \exp \left\{ \int_0^x b(y)w(y)dy \right\} dx \right\} \tag{3} \]

2.2. FCP model

Mean value function \( m_f(t) \) is used to denote the expected number of faults removed till time \( t \) and \( \lambda_f(t) = dm_f(t)/dt \) is used to denote the fault removal intensity function. Since a removed fault must first be detected, FCP can be modeled as a separate process following FDP with a debugging delay. For convenience of discussion, the testing effort consumed during the period from detection of a fault to the final removal of the fault is termed as correction effort of the fault. Generally correcting different faults requires different amounts of testing resources, hence correction effort can be modeled as a random variable with probability density function (pdf) and the cumulative distribution function (cdf) denoted as \( f(x) \) and \( F(x) \).

Thus it can be obtained that

\[ m_f(t) = \int_0^t \lambda_f(y)F(W(t) - W(y))dy \tag{4} \]

where \( F(W(t) - W(y)) \) is the probability that the fault detected at \( y \) is corrected before \( t \).

Different \( m_f(t) \) can be derived based on \( m_d(t) \) and different \( f(x) \). Furthermore, we have

\[ \lambda_f(t) = \int_0^{W(t)} \lambda_d(W^{-1}(W(t) - x))f(x)w(t)dw^{-1}(W(t) - x) \]

\[ = \int_0^{W(t)} \lambda_d(W^{-1}(W(t) - x))f(x)w(t)dx \frac{dW^{-1}(W(t) - x)}{w(W^{-1}(W(t) - x))} \tag{5} \]

Different \( m_f(t) \) can be derived based on \( m_d(t) \) and different \( f(x) \).

3. Some specific models

Fault detection rate function \( b(t) \) is usually assumed to be constant and it is denoted as \( b \) here [21]. From (2) we have

\[ m_d(t) = a(t) - a \exp(-bW^a(t)) - \exp(-bW^a(t)) \int_0^t a'(x) \exp(bW^a(x))dx \tag{6} \]

The total number of faults \( a(t) \) was usually assumed to be an exponential or linear function of time in the literature. Yamada et al. [34] proposed two FDP models with consideration of imperfect debugging, by assuming that the expected total number of faults increases exponentially and linearly with the testing time, respectively. An S-shaped concave FDP model was proposed in Pham et al. [25] assuming that the total number of faults is a linear function of the testing time. In the following subsections various TEF dependent FDP and FCP models are derived based on different assumptions on \( a(t) \) and \( f(x) \).

3.1. Paired model 1

We assume that the total number of faults increases exponentially with the total testing effort consumed and the correction effort required is an exponential variable as

\[ a(t) = a \exp(aW^a(t)), \quad a \geq 0 \tag{7} \]

\[ f(x) = c \exp(-cx) \tag{8} \]

In this case we have

\[ m_d(t) = \frac{ab}{b + a} \left( \exp\{aW^a(t)\} - \exp\{-bW^a(t)\} \right) \tag{9} \]

\[ m_f(t) = \begin{cases} \frac{ab}{b + a} \left( \exp\{aW^a(t)\} + a \exp\{-bW^a(t)\} - \frac{ab}{b + a} (1 + bW^a(t)) \exp\{-bW^a(t)\} \right), & c = b \\ \frac{a}{1 + a} \left( c \exp\{aW^a(t)\} + a \exp\{-cW^a(t)\} - c \exp\{-bW^a(t)\} - b \exp\{-cW^a(t)\} \right), & c \neq b \end{cases} \tag{10} \]
Actually (9) can be obtained by combining (6) and (7), (10) can be obtained by substituting (9) into (3), and (4). When \( W^*(t) = t \), (9) is the same as the FDP model obtained in Yamada et al. [34] for the case when the total number of faults is an exponential function of testing time. When \( \alpha = 0 \) and \( W^*(t) = t \), (9) and (10) are the same as the paired model obtained in Wu et al. [31] for the case of exponential debugging delay.

3.2. Paired model 2

We assume that the total number of faults increases exponentially with the total testing effort consumed as given in (7) and the correction effort required is a gamma variable as

\[
f(x) = \frac{\alpha \exp(-\alpha x) (\mu \epsilon x)^{c-1}}{\Gamma(c)}
\]

where \( \Gamma(c) = \int_0^\infty \exp(-y)y^{c-1} \, dy \) is the Euler gamma function.

Similarly we have

\[
m_d(t) = \frac{ab}{b + a} \exp\{aW^*(t)\} - \exp\{-bW^*(t)\}
\]

\[
m_c(t) = \begin{cases} \frac{a + \exp\{aW^*(t)\} - \exp\{-aW^*(t)\}}{1 + \exp\{aW^*(t)\} - \exp\{-aW^*(t)\}} & \mu = b \\ \frac{a + \exp\{aW^*(t)\} - \exp\{-aW^*(t)\}}{1 + \exp\{aW^*(t)\} - \exp\{-aW^*(t)\}} & \mu \neq b \end{cases}
\]

where \( \Gamma(c_1, c_2, c_3) = \int_0^\infty \exp(-y)y^{c-1} \, dy \) is a generalized incomplete gamma function. When \( \alpha = 0 \) and \( W^*(t) = t \), (12) and (13) are the same as the paired model obtained in Wu et al. [31] for the case of gamma debugging delay.

3.3. Paired model 3

We assume that the total number of faults increases linearly with the total testing effort consumed and the correction effort required is an exponential variable as

\[ a(t) = a + sW^*(t), \quad s \geq 0 \]

\[
f(x) = c \exp(-cx)
\]

In this case we have

\[
m_d(t) = \left( a - \frac{s}{b} \right) (1 - \exp\{-bW^*(t)\}) + sW^*(t)
\]

\[
m_c(t) = \begin{cases} \left( a - \frac{s}{b} \right) (1 - bW^*(t)) \exp\{-bW^*(t)\} + sW^*(t)(1 - \exp\{-bW^*(t)\}) & c = b \\ \left( a - \frac{s}{b} \right) \left( 1 + b \exp\{-cw^*(t)\} - c \exp\{-bw^*(t)\} \right) + sW^*(t) - \frac{s}{b} (1 - \exp\{-cW^*(t)\}) & c \neq b \end{cases}
\]

Actually (16) can be obtained by combining (6) and (14). (17) can be obtained by substituting (16) into (3) and (4). When \( W^*(t) = t \), (16) is the same as the FDP model obtained in Yamada et al. [34] for the case when the total number of faults is a linear function of testing time. When \( s = 0 \) and \( W^*(t) = t \), (16) and (17) are the same as the paired model obtained in Wu et al. [31] for the case of exponential debugging delay.

3.4. Paired model 4

We assume that the total number of faults increases linearly with the total testing effort consumed as given in (14) and the correction effort required is a gamma variable as given in (11).

Similarly we have

\[
m_d(t) = \left( a - \frac{s}{b} \right) (1 - \exp\{-bW^*(t)\}) + sW^*(t)
\]

\[
m_c(t) = \begin{cases} \frac{a + \exp\{aW^*(t)\} - \exp\{-aW^*(t)\}}{1 + \exp\{aW^*(t)\} - \exp\{-aW^*(t)\}} & \mu = b \\ \frac{a + \exp\{aW^*(t)\} - \exp\{-aW^*(t)\}}{1 + \exp\{aW^*(t)\} - \exp\{-aW^*(t)\}} & \mu \neq b \end{cases}
\]

when \( s = 0 \) and \( W^*(t) = t \), (18) and (19) are the same as the paired model obtained in Wu et al. [31] for the case of gamma debugging delay.

4. A summary of various testing effort functions

Testing effort functions that have been commonly used include Constant, Exponential, Rayleigh, Weibull and Logistic curves. Exponential curve and Rayleigh curve can be regarded as special cases of Weibull curve. The details are shown below.

4.1. Constant TEF

We assume that \( w(t) \) is a constant. It can be expressed as

\[ w(t) = w \]

Thus the cumulative testing effort \( W(t) \) can be obtained as

\[ W(t) = wt \]

It can be seen that the total testing effort consumed tends to positive infinity when \( t \) approaches positive infinity. In the case that TEF is not considered, it can be regarded as considering \( w(t) = 1 \).

4.2. Weibull TEF

Weibull TEF is very flexible and it can well fit most data that are often used in the study of SRGM. The cumulative testing effort \( W(t) \) is given by

\[ W(t) = N(1 - \exp(-\beta t^m)) \]

where \( N \) is the expected total amount of testing effort that is required by software testing, \( \beta \) and \( m \) are the scale parameter and shape parameter, respectively. It should also be noted that the cumulative testing effort consumed is finite and tends to \( N \) when \( t \) approaches positive infinity.

Differentiating (22) gives

\[ w(t) = N \beta mt^{m-1} \exp(-\beta t^m) \]

The exponential TEF is a special case of Weibull TEF when \( m = 1 \). Exponential curve is suitable to describe the testing environment which has a monotonically declining testing effort rate.

The Rayleigh TEF is a special case of Weibull TEF when \( m = 2 \). Rayleigh testing effort rate first increases to its peak, then decreases with a decelerating speed to zero asymptotically without reaching zero.
4.3. Logistic TEF

Logistic curve was first proposed in Parr [24] as an alternative of Rayleigh curve. It exhibits similar behavior as Rayleigh curve, except during the initial stage of the project. The logistic cumulative testing effort $W(t)$ is given by

$$W(t) = \frac{N}{1 + A e^{-\eta t}}$$

(24)

where $A$ is a constant parameter and $\eta$ is the consumption rate of testing effort expenditure. Similar to the Weibull case, the cumulative testing effort consumed is finite and tends to $N$ when $t$ approaches positive infinity.

Taking derivatives on both sides of (24) gives

$$w(t) = \frac{NA_0}{(e^\eta t^2 + A e^{-\eta t})^2}$$

(25)

where $w(t)$ is the testing effort rate. The logistic curve was first proposed in Parr[24] as an alternative of Rayleigh curve. It exhibits similar behavior as Rayleigh curve, except during the initial stage of the project. The logistic cumulative testing effort consumed is finite and tends to $N$ when $t$ approaches positive infinity.

$$w(t) = \frac{NA_0}{(e^\eta t^2 + A e^{-\eta t})^2}$$

(25)

where $w(t)$ is the testing effort rate. The logistic curve was first proposed in Parr[24] as an alternative of Rayleigh curve. It exhibits similar behavior as Rayleigh curve, except during the initial stage of the project. The logistic cumulative testing effort consumed is finite and tends to $N$ when $t$ approaches positive infinity.

5. Illustrative example

5.1. Dataset description

The dataset we use is from the System T1 data of the Rome Air Development Center (RADC) [22]. Although this is quite an old dataset, some criteria are used to compare the performances of different TEFs.

Taking derivatives on both sides of (24) gives

$$w(t) = \frac{NA_0}{(e^\eta t^2 + A e^{-\eta t})^2}$$

(25)

where $w(t)$ is the testing effort rate. The logistic curve was first proposed in Parr[24] as an alternative of Rayleigh curve. It exhibits similar behavior as Rayleigh curve, except during the initial stage of the project. The logistic cumulative testing effort consumed is finite and tends to $N$ when $t$ approaches positive infinity.

$$w(t) = \frac{NA_0}{(e^\eta t^2 + A e^{-\eta t})^2}$$

(25)

where $w(t)$ is the testing effort rate. The logistic curve was first proposed in Parr[24] as an alternative of Rayleigh curve. It exhibits similar behavior as Rayleigh curve, except during the initial stage of the project. The logistic cumulative testing effort consumed is finite and tends to $N$ when $t$ approaches positive infinity.

5.2. Select the most suitable TEF for this dataset

Parameters in the different types of TEF are estimated by Least Square Error (LSE). In order to select a TEF that best fits this dataset, some criteria are used to compare the performances of different TEFs.

(1) RMSE

The Root Mean Square Error (RMSE) is defined as

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (w_i - \hat{w}_i)^2}$$

(26)

A smaller RMSE indicates a smaller fitting error and better performance.

(2) Bias

The bias is defined as the sum of the deviation of the estimated testing curve from the actual data, as shown below:

$$Bias = \frac{1}{n} \sum_{i=1}^{n} (w_i - \hat{w}_i)$$

(27)

(3) Variance

The variance is defined as [8]

$$Variance = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (w_i - \hat{w}_i - Bias)^2}$$

(28)

(4) RMSPE

The Root Mean Square Prediction Error (RMSPE) is defined as [8]

$$RMSPE = \sqrt{Variance + Bias^2}$$

(29)

RMSPE is also a measure to depict how close the model predicts the observation.

Estimated parameters and comparison results for different TEFs are shown in Table 2. Fig. 1 is plotted for graphical illustration.

It can be seen that logistic TEF has the smallest RMSE, Variance, and RMSPE and also has a smaller Bias than Weibull TEF. Fig. 1 also shows that logistic TEF fits best. Thus logistic TEF is adopted for further analysis.

5.3. Performance analysis

The paired model (9) and (10) is used for illustration. After substituting the cumulative logistic testing effort function (24) with the estimated parameters $N=321.482$, $\eta=0.3826$, and $A=423.788$ into (9) and (10), the paired model is applied to fit against the real dataset. The LSE estimation of the parameters are obtained as $\hat{a}=100.97$, $\hat{b}=0.0094$, $\hat{c}=0.0418$. According to the estimated parameters, there are about 100.97 faults at the beginning of testing. The total number of faults when $t$ approaches infinity is expected to be $lim_{t \to \infty} \hat{a}(t) = 198.01$. Fig. 2 is plotted for graphical illustration.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>The dataset – System T1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weeks</td>
<td>Computer time (CPU hours)</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4.3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0.6</td>
</tr>
<tr>
<td>5</td>
<td>2.3</td>
</tr>
<tr>
<td>6</td>
<td>1.6</td>
</tr>
<tr>
<td>7</td>
<td>1.8</td>
</tr>
<tr>
<td>8</td>
<td>14.7</td>
</tr>
<tr>
<td>9</td>
<td>25.1</td>
</tr>
<tr>
<td>10</td>
<td>4.5</td>
</tr>
<tr>
<td>11</td>
<td>9.5</td>
</tr>
<tr>
<td>12</td>
<td>8.5</td>
</tr>
<tr>
<td>13</td>
<td>29.5</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
</tr>
<tr>
<td>15</td>
<td>39.5</td>
</tr>
<tr>
<td>16</td>
<td>26</td>
</tr>
<tr>
<td>17</td>
<td>25</td>
</tr>
<tr>
<td>18</td>
<td>31.4</td>
</tr>
<tr>
<td>19</td>
<td>30</td>
</tr>
<tr>
<td>20</td>
<td>12</td>
</tr>
<tr>
<td>21</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Estimated parameters and comparison results for different TEF.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEF</td>
<td>Estimated parameters</td>
</tr>
<tr>
<td>Constant</td>
<td>$w=14.29$</td>
</tr>
<tr>
<td>Weibull</td>
<td>$N=407.0830$, $\beta=2.064-4$, $m=2.923$</td>
</tr>
<tr>
<td>Logistic</td>
<td>$N=321.482$, $\eta=0.3826$, $A=423.788$</td>
</tr>
</tbody>
</table>
6. Software release policies

Determination of the optimal release time is a critical decision for software projects and has been studied in many papers [1, 11, 20]. As cost and reliability requirements are of great concern, they are often used to determine the time to stop the testing and release the software [21, 27].

6.1. Software release policy based on reliability criterion

Software reliability is defined as the probability that no failure occurs during time interval \([T, T+\Delta T]\) given that the software is released at time \(T\). Considering that software normally does not change in operational phase, the reliability function is

\[
R(\Delta T|T) = \exp(-\lambda_d(\Delta T)) \tag{30}
\]

If \(R_1\) is the reliability target and \(T_{LC}\) is the length of the software life cycle, the time when the reliability of the software reaches \(R_1\) can be obtained as \(T_1 = \inf(\lambda_d(T) \leq \ln(1/R_1)/\Delta T : T \in [0, T_{LC}])\).

6.2. Software release policy based on cost criterion

Besides the reliability requirement, we can also discuss the optimal release time based on the total cost during the software testing phase and the operational phase. With the incorporation of FCP \(m_d(t)\), the cost model can be expressed as

\[
C(T) = c_1 m_d(T) + c_2 (m_d(T_{LC}) - m_d(T)) + c_3 W^*(T) \tag{31}
\]

where \(c_1\) is the cost of fixing a fault during the testing phase, \(c_2\) is the cost of fixing a fault during the operational phase \((c_2 > c_1 > 0)\), \(c_3\) is the unit cost for testing effort consumed during testing. By minimizing the cost model with respect to \(T\), the optimal release time \(T_c\) can be obtained.

Differentiating both sides of (31), we have

\[
C'(T) = c_3 W(T) - (c_2 - c_1) \lambda_d(T) \tag{32}
\]

Furthermore we have \(C'(0) = c_3 W(0) > 0\). Let \(z_1 \leq z_2 \leq \ldots \leq z_n\) be all the solutions to \(\lambda_d(T)/W(T) = c_3/(c_2 - c_1)\) during \((0, T_{LC})\). If \(n = 2k\) \((k \geq 0)\), \(T_c\) can be determined as \(T_c = \arg \min_{T} = T_{z_k} C(T)\). Otherwise \(n = 2k + 1\) and \(T_c\) can be determined as \(T_c = \arg \min_{T} = T_{z_k} C(T)\).

6.3. Software release policy based on mixed criterion

When both reliability requirements and the total cost are considered, our goal is to determine the optimal release time \(T^*\) which minimizes the total cost without compromising the reliability requirements. Thus the problem can be formulated as

Minimize \(C(T) = c_1 m_d(T) + c_3 (m_d(T_{LC}) - m_d(T)) + c_3 W^*(T)\)

Subject to \(R(\Delta T|T) = \exp(-\lambda_d(\Delta T)) \geq R_1\)

The time axis \([T_1, T_{LC}]\) can be divided into four types of intervals such that both \(R(\Delta T|T)\) and \(C(T)\) increase on type 1 intervals, both \(R(\Delta T|T)\) and \(C(T)\) decrease on type 2 intervals, \(R(\Delta T|T)\) increases while \(C(T)\) decreases on type 3 intervals, and \(R(\Delta T|T)\) decreases while \(C(T)\) increases on type 4 intervals. The candidates for \(T^*\) comprise of the minimum \(T\) in each type 1 interval that satisfies \(R(\Delta T|T) \geq R_1\), the maximum \(T\) in each type 2 interval that satisfies \(R(\Delta T|T) \geq R_1\), the end points of type 3 intervals which satisfy \(R(\Delta T|T)\)
\[ \geq R_1, \text{ and the initial points of type 4 intervals which satisfy } R(\Delta T) \geq R_1, \text{ } T^* \text{ equals to the candidate which incurs the lowest cost.} \]

### 6.4. Numerical examples for software release policy

For illustration, we consider the first paired model (9) and (10) with parameters estimated as \( a = 100.97, b = 0.0094, c = 0.0021 \) and \( c = 0.0418 \) and logistic TEF with parameters estimated as \( N = 321.482, \eta = 0.3826, \) and \( A = 423.788. \) We also assume \( T_{\text{c}} = 300, \) \( c_1 = 300, \) \( c_2 = 2000, \) \( c_3 = 700, \) \( \Delta T = 10, \) and \( R_1 = 0.95. \)

From (9) and (10) we have

\[ \lambda(T) = b w(T) \left( \frac{a}{b - a} \exp\left( c W^*(T) \right) + \frac{ab}{b - a} \exp\left( -b W^*(T) \right) \right) = \frac{9033.4 \exp(0.0021 W^*(T)) + 40439 \exp(-0.0094 W^*(T))}{(0.1913 T) + 423.788 \exp(-0.1913 T)^s}. \]

(33)

\[ \text{It can be seen that } \lambda(T) \text{ increases from } [0, 14.112] \text{ and on } (14.112, 300). \text{ Solving } \lambda(T) = \ln(T / R_1) / \Delta T = 0.0051 \text{ gives } T_1 = 39.626. \text{ The reliability requirement is satisfied if the software is released after 39.626 weeks of testing.} \]

From (46) and (47) we have

\[ C(T) = C_1(T) + C_2(T) \]

\[ = \frac{253590 - 1700 m_1(T) + 700 W^*(T)}{700 w(T) - 1700 c_1(T)} \]

(34)

(34)

In addition we have

\[ \lambda(T) = \frac{100.97}{b + c} \exp(c w(T)) - \exp(-c W^*(T)) \]

\[ + \frac{1.0009 \exp(0.0021 W^*(T)) - 1.1659 \exp(-0.0418 W^*(T))}{(b + c) (b - c)} \]

\[ = 0.165 \exp(0.0021 W^*(T)) - 1.1659 \exp(-0.0418 W^*(T)) + 1.0009 \exp(-0.0094 W^*(T)) \]

(35)

Solving \( \lambda(T)/w(T) = 700/1700 \) gives \( T = 8.013 \) and 16.848. Thus \( C(T) \) increases on \( [0, 8.013] \), decreases on \( (8.013, 16.848) \) and increases on \( [16.848, 300]. \) The optimal release time which minimizes the total cost is \( T_1 = 16.848. \) The corresponding total cost is \( C(T) = 5271820. \)

As both \( R(\Delta T) \) and \( C(T) \) increase on \( [T_1, 300] \), the optimal software release time \( T^* = T_1 = 39.626. \) Fig. 3 is plotted for graphical illustration.

### 7. Conclusions

This paper studies testing effort function dependent software FDP and FCP with incorporation of imperfect debugging. Testing resource is usually not constantly allocated during software testing phase, which can largely influence the fault detection rate and the time needed to correct the detected faults. For example, the debugger may spend a week without doing any testing work and work very hard in the following few days. In addition, it is natural for debuggers to make mistakes and introduce new faults during testing. The debuggers tend to introduce more faults if more testing effort is consumed since the code has experienced more changes. In order to capture the influences of testing resource allocation and fault introduction on both FDP and FCP, we first derive FDP incorporating testing effort function and the fault introduction effect, and then obtain FCP as delayed FDP with a correction effect. Various paired FDP and FCP models are obtained based on different assumptions on fault introduction and correction effect. It can be seen that our model is quite general and flexible. Some simpler models are the special cases of our models. An example is presented to illustrate the application of the paired models. The optimal release policy under different criteria is also studied.

### Acknowledgments

The research report here was partially supported by the NSFC under Grant nos. 71231001 and 71301009, China Postdoctoral Science Foundation funded project under Grant no. 2013M530531, and by the MOE Ph.D. supervisor fund, 20120006110025.

### References


