Optimal Allocation Based on Cost, Testing Effort and Reliability

Nan ZHANG
Harbin University of Commerce, Harbin 150028, China

Keywords: Software reliability, Testing effort, Testing resource, Optimal allocation.

Abstract. In this paper, we investigate a software resource allocation problem. We incorporate generalized modified Weibull testing effort function in software reliability growth model to get a better description of the software debugging process. The main purpose is to minimize the cost of software development when the number of remaining faults and a desired reliability objective are given. A real software failure project is demonstrated the effectiveness of proposed models.

Notation

- \( \alpha \) the expected number of initial faults
- \( b \) the rate of fault detection per testing effort at time \( t \)
- \( m(t) \) the expected mean number of faults detected in time \( (0, t) \)
- \( w(t) \) the current testing effort consumption at time \( t \)
- \( W(t) \) the cumulative testing effort consumption in time \( (0, t) \)
- \( \alpha \) the total amount of testing effort
- \( \beta \) a scale parameter
- \( m \) a shape parameter
- \( \theta \) a shape parameter
- \( \lambda \) a kind of accelerating factor parameter
- \( v_i \) a weighting factor to measure the relative importance of a fault correction from module \( i \).
- \( C_1 \) cost of a fault removed during debugging process.
- \( C_2 \) cost of a fault removed during operation process.
- \( C_3 \) cost of testing per unit testing effort expenditure.

Introduction

Software reliability is one of important index in software development process, which defined as the probability of failure-free software operation in a specified time and space. Since 1970s, software reliability growth models have been built to estimate software reliability during software development processes [1-2]. In these software reliability growth models, an exponential and an S-shaped curve are described to fault phenomenon in software debugging process [1-2]. Besides, in these models, the span of time concludes calendar time, clock time, and CPU execution time. Moreover, some researchers verified that the CPU execution time is a better time for software reliability growth models than other time [3-5].

Specification, design, coding, and testing are composed of a software development process. The testing is the last stage of software development, where the software product is tested to debug software faults and is confirmed the software reliability to achieve the requirements of users. When software resources available are limited, it is important to allocate the testing resources so that the maximum of software reliability is reached. As a fact, the reliability of the software is indirectly proportional to the number of faults corrected. The problem of maximization of the software reliability may be treated as an equivalent problem of the maximization of number of faults to be removed. Many authors have addressed this problem to determine the optimal resource allocation over the years, including [6-8].
The remainder of paper is organized as follows. We incorporate generalized modified weibull testing effort function into the software reliability growth model in section 2. The applications of these models to actual data set are discussed in section 3. Section 5 is conclusions.

Software Reliability Growth Models with Generalized Modified Weibull Testing Effort Function

The distribution of resource consumed in the software debugging process is defined as a consumption curve of software testing effort [9]. The testing-effort can be represented as the number of CPU hours, the number of executed test cases, etc. Furthermore, experience has studied that testing effort are usually performed by different curve in the software debugging phase of a project [9].

Let \( W(t) \) be the cumulative amount of testing effort expenditures in the testing time interval \((0, t]\) and \( f(t) \) be the consumption rate of the testing effort expenditures. According to the assumptions, we get the different equation:

\[
\frac{dW(t)}{dt} = f(t)[\alpha - W(t)]
\]

Solving equation (1), we get

\[
W(t) = \int_0^t w(x)dx
\]

By various values to \( w(t) \), we obtain different testing effort function models.

In this paper, a new software reliability growth model with the generalized modified weibull testing effort function is proposed to predict the software reliability. Here are assumptions in model:

1. Software debugging process is modeled with a Non-Homogeneous Poisson Process.
2. In the time \( t \), the mean number of faults detected by testing effort expenditures is proportional to the mean number of remaining faults
3. Testing effort expenditures are described by a generalized modified weibull curves.
4. The new models are perfect debugging.

Based on assumptions, we can get the following differential equation [1-2]

\[
\frac{dm(t)}{dt} + \frac{1}{w(t)} = b[a - m(t)]
\]

Solving the above equation under boundary condition \( m(0) = 0 \), we get

\[
m(t) = a[1 - \exp(-bW(t) + bW(0))]
\]

The generalized modified weibull distribution with five parameters had a great flexibility in accommodating all the forms of the hazard rate function, can be used in a variety of problems for modeling software faults [3-5]. The generalized modified weibull testing effort function is described as follow:

\[
W(t) = \alpha[1 - \exp(-\beta t^\gamma \exp(-\lambda t))]^\theta
\]

The fault debugging process is modeled by a Non-Homogeneous Poisson Process. The probability of failure-free software operation in a specified time and space is given by

\[
R(t) \equiv R(t + \Delta t | t) = \exp\{-m(t + \Delta) - m(t)\}
\]

Software reliability is also defined as the ratio of the cumulative number of fault detected at time \( t \) to the expected number of initial faults [1-2].

\[
R(t) = \frac{m(t)}{a}
\]

Through above equation (7), we can determine a unique value of \( t \).
Resource Allocation

This section investigated resource allocation problem based on a Software reliability growth model with generalized modified Weibull testing effort function. Assumptions as follow:

1. The software system consisted of $N$ independent modules.
2. The each module is estimated by a Software reliability growth model with generalized modified Weibull testing effort function in debugging process.
3. The expenditures of testing resource in debugging process are denoted by $W$.
4. In minimized number of remained faults, the testing resource $W$ is allocated to each module.

According to the above assumptions, the mean value function is given by

$$M(t) = \sum_{i=1}^{N} v_i m_i(t) = \sum_{i=1}^{N} v_i a_i [1 - \exp(-b_i W_i(t))]$$  \hspace{1cm} (8)

Many cost models are proposed in the literature, such as RCA PRICE S model, Putnam’s SLIM cost model, Boehm’s COCOMO model, Checkpoint, COOMO’II and so on [6, 10]. In this paper, software cost is evaluated by testing effort expenditures, cost criterion, and cost of removal faults. The equation as follow:

$$C(t) = C_1 m(t) + C_2 [m(t_{\infty}) - m(t)] + C_3 \int_{t_0}^{t} w(x)dx$$  \hspace{1cm} (9)

According to equation (4), equation (8), and equation (9), a software cost model based on Non-Homogeneous Poisson Process is computed by:

$$\sum_{i=1}^{N} \text{cost}_i(W_i) = C_1 \sum_{i=1}^{N} v_i a_i [1 - \exp(-b_i W_i)] + C_2 \sum_{i=1}^{N} v_i a_i \exp(-b_i W_i) + C_3 \sum_{i=1}^{N} W_i$$  \hspace{1cm} (10)

From equation (10), we can see the relationship between software cost and testing resource. We supported that denoted the amount of testing effort and $C_i$ is the testing effort of module. Therefore, the optimization equation is given by:

\[
\begin{cases}
\text{Minimize: } \sum_{i=1}^{N} \text{Cost}_i(W_i) \\
\sum_{i=1}^{N} W_i \leq W, W_i \geq 0, i = 1, 2, \cdots, N \\
R_i(t) = 1 - \exp(-b_i W_i(t)) \geq R_0
\end{cases}
\]  \hspace{1cm} (11)

Solving equation (11), we get

$$W_i \geq -\frac{1}{b_i} \ln(1 - R_0)$$  \hspace{1cm} (12)

Moreover, we have

\[
\begin{cases}
\sum_{i=1}^{N} W_i \leq W, W_i \geq C_i, W_i \geq 0, i = 1, 2, \cdots, N \\
C_i = \max(0, -\frac{1}{b_i} \ln(1 - R_0), -\frac{1}{b_i} \ln(1 - R_0), \cdots, -\frac{1}{b_i} \ln(1 - R_0))
\end{cases}
\]  \hspace{1cm} (13)

Here, let $X_i = W_i - C_i$, equation (13) transforms to:
Solving above equation (14) [6], we can obtain:

\[
\sum_{i=1}^{N} \text{cost}_i(W_i) = C_1 \sum_{i=1}^{N} v_i a_i [1 - \exp(-b_i C_i) \exp(-b_i X_i)] + C_2 \sum_{i=1}^{N} v_i a_i \exp(-b_i C_i) \exp(-b_i X_i) + C_3 \sum_{i=1}^{N} (X_i + C_i)
\]

**(Numerical Examples)**

In section, we evaluate the performance of the proposed models based on the real software faults data set. The method of least square estimation minimizes the sum of squares of the deviations between what we actually observe, and what we expect. Sometimes least square estimation may have smaller bias or it may approach normality faster. Thus, we use the method of least square estimation to estimate the parameters of the proposed models [11-13].

Assume that a software system is composed of 6 modules, and the total amount of generalized modified Weibull testing effort expenditures is 50000 man-hours. The software reliability is 0.95. According to above equations, the optimal expenditures of testing effort can be estimated and shown in table 1.

<table>
<thead>
<tr>
<th>Module</th>
<th>ai</th>
<th>bi</th>
<th>Wi</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23</td>
<td>0.00139</td>
<td>2997.1</td>
<td>1572.7</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>0.000912</td>
<td>3721.6</td>
<td>2002.8</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0.00603</td>
<td>4016.7</td>
<td>2113.7</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>0.000598</td>
<td>8213.9</td>
<td>4165.6</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>0.000269</td>
<td>24403.5</td>
<td>12168.3</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>0.000427</td>
<td>6648.2</td>
<td>3401.2</td>
</tr>
</tbody>
</table>

**Conclusions**

In this paper, we incorporated generalized modified Weibull testing effort into software reliability growth models. Besides, this paper proposed a method to optimize allocation of testing resource considering cost, reliability and testing effort. Numerical examples are described and discussed.

**Acknowledgement**

This paper is supported by the Heilongjiang Province postdoctoral Grant No. LBH-Z15115, and the Harbin University of Commerce under Grant No. 15RW21.

**References**


