Residual Fault Detection and Performance Analysis of G-O Software Growth Model

Chandra Mouli Venkata Srinivas Akana, Dr. C. Divakar, Dr. Ch. Satyanarayana

Research Scholar, Dept. of CSE, JNTUK, Kakinada, Andhra Pradesh, India.

Principal, Avanthi’s Research and Technological Academy, Vijayanagaram, AP, India.

Professor, Dept of Computer Science and Engineering, JNTUK, Kakinada, AP, India.

Email: mouliac@yahoo.co.in

DOI: http://dx.doi.org/10.15520%2Fjccst.2015.vol5.iss03.44

Abstract: Quality assessment for the software growth models (SRGMs) is crucial in deciding the growth model for the application. The main goal of the growth model is to provide higher efficiency with optimal reliability. So there should be an analytical approach for the measurement of these factors. Here in this paper we propose one such exponential approach for the assessment of G-O (Goel, Okumoto) growth model under certain residual faults based on NHPP.

Keywords: Software reliability, G-O model, Quality assessment fault detection and residual faults

INTRODUCTION

A large number of software reliability growth models have been proposed so far to analyze the reliability of growth models during software application testing. There is a high demand to deliver high-quality software, more accurate software reliability models are required to estimate the optimal software release time and the cost of testing efforts. Software reliability Engineering (SRE) is the discipline that helps the organizations to improve the quality of their products and processes. The main attributes of software growth model quality which is generally accepted as one of the major factors in software quality assessment since it quantifies the failures. Finally we can conclude that a quality software model which depends on the focused software is needed to be successfully applied for different systems. This model attempt to match product properties with the software quality attributes. Software reliability which is defined as the probability of failure-free software operation for a specified period of time in a specified environment [1] [2] is the most important quality metric. Software reliability and the cost of software development have become a crucial standard of software quality. The cost of software development is inseparable with the estimation of software reliability. So the accurate and concise estimation of software reliability can minimize the cost of software development and get more profit. With the increasing demand to deliver high-quality software, more accurate software reliability models are required to estimate the optimal software release time and the cost of testing efforts. In this paper we discuss a concise and accurate G-O model based on NHPP.

In this paper we focus the observation on G-O growth model. The paper is organized as follows, section II discusses about the G-O SRGM and its functionalities, section III explains about the fault detection and its analysis. Section IV briefly gives a mathematical analysis for the performance analysis of the proposed approach ending with the experimental results and conclusions.

GOEL-OKUMOTO GROWTH MODEL

In this G-O growth model is considered for the quality assessment.

The following are the assumptions are considered for the G-O model

a. G-O model is NHPP (Non Homogenous Poisson process)

b. Value of fault detection rate is constant;

c. Each software failure and fault removal occurs at independently

d. Each time a failure occurs, the error which caused it is immediately removed, and no other errors are introduced

e. The probability of failure occurring is proportional to the number of residual faults which are not yet observed

The probability of fault detection is proportional to the number of residual faults which are not yet observed.

From the above considerations the G-O model can be formulated as

$$\frac{dm(t)}{dt} = b(a - m(t))$$ (1)

When the above equation is solved

$$m(t) = a(1 - e^{-bt})$$ (2)

Where 'a' denotes the numbers of faults are to be eventually detected, it concludes the introduction of new faults during debugging a fault. Parameter 'b' denotes the fault detection rate. This presents that the probability of every fault detected is invariable every time.

FAULT DETECTION ANALYSIS

Fault removal and insertion is a random process [3]. So here we assume that insertion of new fault during the removal process is β.

The above equation can be modeled as
\[ m'(t) = \frac{a}{1 \pm b} \left( 1 - e^{-(1 \pm b)t} \right) \]  
\[ \text{Where } a' = \frac{a}{1 \pm b} \text{ and } b' = (1 \pm b) \phi \]  
So it can be rewritten as
\[ m'(t) = a' \left( 1 - e^{-b't} \right) \]  
From the above (2) & (4) both \( m(t) \) and \( m'(t) \) are same, so it has no critical effect on accuracy of reliability assessment.

**EFFICIENCY ANALYSIS**

Let us assume the efficiency as ‘ \( \square \) ’ equation (1) can be written as
\[ \frac{\partial m}{\partial t} = b(a - \epsilon m(t)) \]  
By solving the above equations we get
\[ m'(t) = \frac{b}{a - b} \left( 1 - e^{-(1 - b)t} \right) \]  
Assuming \( a' = \frac{a}{1 - b} \) and \( b' = (1 - b) \phi \)  
So
\[ m'(t) = a' \left( 1 - e^{-b't} \right) \]  
The fault removal efficiency has no critical effect on accuracy of reliability prediction of this model.

**PROPOSED METHOD**

Fault introduction and its removal efficiency will vary with sample and doesn’t have any critical effect on the accuracy of reliability estimation with respect to the G-O model. Instead the parameters \( b \), which indicates the fault detection rate during testing phase, plays a significant role to guarantee the accuracy of software reliability growth modeling. From the studies, it is evident that one can estimate the testing efforts consumed in testing phase and predict the trends of the fault detection rate. The fault detection rate is not constant. During the process of software system running, the software testers will be gradually familiar with the software system with the time, which has a positive effect on the value of fault detection rate; On the other hand, with the increase of testing time the number of residual faults is less and less. Then it will be gradually difficult to detect the faults in software system. So the time dependent fault detection rate should be estimated considering the two aspects above jointly. We suggest the proposed model has the following explicit assumptions:

a. Failure observation and fault removal phenomenon is modeled by NHPP.

b. Testers’ learning capability is a non-decreasing function which is relevant to testing time [4][5][6].

c. Each software failure and fault removal occurs at independently.

d. The detection rate of residual faults in software system is a non-increasing time dependent function

e. The probability of failure occurring is proportional to the number of residual faults which are not yet observed.

f. The probability of fault detection is proportional to the number of residual faults which are not yet observed.

According to the several assumptions above we can construct a differential equation of G-O model as follows

\[
\frac{dn(s)}{ds} = b(t)(a - m(t)) \]

Since the parameter \( b \) is between 0 and 1 and \( \phi \) is a time dependent function and it is estimated by testers learning ability and the number of residual faults in software system.

\[
\begin{align*}
b(t) &= 1 - (1 - b)e^{-\epsilon t} \\
b(t) &= b(0) - b e^{-\epsilon t}
\end{align*}
\]

From the above two equations \( b(t) \) is the fault detection rate incorporating the rise of tester’s learning ability and the \( b(t) \) is the fault detection rate considering the remaining faults during time \( t \) in the software system.

**EXPERIMENTAL ANALYSIS**

The performance is measured in terms of sum of square errors which is gradually small when compared with other models.

Sum of squares error (SSE) is used to describe the distance between actual and estimated number of faults detected totally which is defined as

\[
sse = \sum (y - \hat{y})^2
\]

Where ‘\( n \)’ denotes the number of failure samples in the given data set and ‘\( t \)’ the interval. The model can provide a better goodness of fit when the value of SSE is smaller.

**Table 1: Data set taken from [7]**

<table>
<thead>
<tr>
<th>Test period</th>
<th>CPU hrs</th>
<th>Defects found</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>520</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>968</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>1430</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>1893</td>
<td>33</td>
</tr>
<tr>
<td>5</td>
<td>2490</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>3058</td>
<td>49</td>
</tr>
<tr>
<td>7</td>
<td>3623</td>
<td>54</td>
</tr>
<tr>
<td>8</td>
<td>4422</td>
<td>58</td>
</tr>
<tr>
<td>9</td>
<td>5218</td>
<td>69</td>
</tr>
<tr>
<td>10</td>
<td>5823</td>
<td>75</td>
</tr>
<tr>
<td>11</td>
<td>6539</td>
<td>81</td>
</tr>
<tr>
<td>12</td>
<td>7083</td>
<td>86</td>
</tr>
<tr>
<td>13</td>
<td>7487</td>
<td>90</td>
</tr>
<tr>
<td>14</td>
<td>7846</td>
<td>93</td>
</tr>
<tr>
<td>15</td>
<td>8205</td>
<td>96</td>
</tr>
<tr>
<td>16</td>
<td>8564</td>
<td>98</td>
</tr>
<tr>
<td>17</td>
<td>8923</td>
<td>99</td>
</tr>
<tr>
<td>18</td>
<td>9282</td>
<td>100</td>
</tr>
<tr>
<td>19</td>
<td>9641</td>
<td>100</td>
</tr>
<tr>
<td>20</td>
<td>10000</td>
<td>100</td>
</tr>
</tbody>
</table>
The set of software faults analyzed here was obtained from literature [7]. In this paper we test the performance of the proposed model by using the data in table 1. The growth curve is plotted in figure 1 where the no of detected faults for the actual data is S-shaped. This presents that in the initial stages of software testing the number of faults is so big that tester’s learning ability dominates the fault detection rate. And in later period of software testing it is more and more difficult to find the residual faults in software system, so the number of residual faults starts to dominate the fault detection rate instead of tester’s learning process. This situation is more consistent with the actual circumstance and we get more realistic experiment result.

![Figure 1: Number faults detected during a span of time ‘t’](image)

<table>
<thead>
<tr>
<th>Model</th>
<th>SSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-O model</td>
<td>155.5</td>
</tr>
<tr>
<td>S-model</td>
<td>325</td>
</tr>
<tr>
<td>proposal</td>
<td>147.2</td>
</tr>
</tbody>
</table>

From the table 2 it can be seen that the value of SSE is smaller in the experiment of the proposed new model compared with other models. The results indicate that our NHPP model based on fault detection rate fits the data in table 1the best and predicts the number of residual faults in software most accurately. Moreover, Bell-SRGM model also took into account the human’s learning ability and the number of residual faults in software system. In this paper the integral of the function can be directly obtained rather than transforming it into question of finding the sum. So the proposed model can achieve a better accuracy.

**CONCLUSION**

Estimation SRGMs release time can decrease the time and cost for testing [8] and it also helps to determine the various resources needed to achieve the efficiency for the given requirements. So more accurate model is needed to decrease the testing cost and increase the profit of releasing software [9] [10]. In this paper we demonstrate that the improved G-O model doesn’t need to consider imperfect debugging, and then a time-dependent fault detection rate model is presented. In this model the fault detection rate is calculated with the number of faults remaining in the software and the human’s learning ability. Considering the two factors jointly the fault detection rate is more realistic and accurate. Moreover, we have discussed the performances of our new SRGM by using actual software failure data. The experiment result shows the new model can provide a better goodness-of-fit compared with other models.

In future there is need to check the validity and effectiveness of proposal and SRGMs developed under the modeling framework by using many actual data.

**REFERENCES**


