SOFTWARE RELIABILITY GROWTH MODEL WITH LOGISTIC TESTING-EFFORT FUNCTION CONSIDERING LOG-LOGISTIC TESTING-EFFORT AND IMPERFECT DEBUGGING

O. Naga Raju¹ and Ch. Satyanaraya²

¹Department of Computer Science & Engineering, Acharya Nagarjuna University, A.P., India.
²Department of Computer Science & Engineering, JNT University, Kakinda, India.
E-mail: onrajunrt@gmail.com

ABSTRACT
In this paper, we investigate a Software Reliability Growth Model (SRGM) based on the Non Homogeneous Poisson Process (NHPP) which incorporates a logistic testing-effort function. We present Software Reliability Growth Model (SRGM) based on non-homogeneous Poisson process (NHPP), which incorporates the amount of testing effort consumptions during software testing phase. The time dependent behavior of testing effort consumptions is described by Log-Logistic curve. [1] Has used this model into SRGM for finite failure NHPP. In this paper, we will show that a Log-Logistic Test-Effort Function (TEF) can be expressed as a Software Development/test-effort curve. It is assume that the error detection rate to the amount of testing-effort spent during the testing phase is proportional to the current error content. The SRGM parameters are estimated by least square estimation (LSE) and Maximum likelihood Estimation (MLE) methods. The method of data analysis for software reliability measurements are presented for three real data set and results are compared with other existing models to show that the proposed model is good enough to give more accurate description of resources consumption give better fit.

Keywords: SRGM, NHHP, Log-logistic TEF, LSE, MLE, Testing Effort Consumptions.

1. INTRODUCTION
A Computer System consists of two major components: hardware and software. Although extensive research has been done in the area of hardware reliability, research has also been conducted to study the software reliability since 1970. Software reliability is the probability of failure-free operation of computer programme for a specified time in a specified environment. Hence, software reliability is a key factor in software development process and represents a customer-oriented view of software quality. It relates to practical operation rather than simply the design of program. Therefore, it is dynamic rather than static.

A common approach for measuring software reliability is by using an analytic model whose parameters are generally estimated from available data on software failure. However, research activities in SRE have been conducted over past 2 decades extensively, and many SRGM have been proposed [2]. SRGM are successful for estimating software reliability and the number of fault remaining in the software system. They can be used to evaluate software development (SD) status and SRE technology quantitatively [3].

In the area of software reliability modeling, SD effort was often described by the traditional exponential, Weibull, Rayleigh or logistic curves [4], [5], [6] and [7]. However, in many software testing environments it is difficult to describe the testing-effort function by the above four effort consumption curves only.

This paper presents SRGM based on NHPP which incorporate the amount of testing effort consumptions during software testing phase. The time dependent behavior of testing effort consumptions is described by Log-Logistic curve. [1] Has used this model into SRGM for finite failure NHPP. We are using it as a test-effort function into SRGM. Maximum likelihood estimation (MLE) and Least square estimation (LSE) methods are used to estimates the parameters. Experiments have been performed based on three real test/debug data sets. Compressions of predictive capabilities between various models are presented. The results show that the SRGMs with a Log-Logistics TEF can estimate the number of initial faults better that the previous approaches.

SRGMs proposed by most papers incorporate the effect of testing effort in the software reliability growth and the software development effort can be described by the traditional Rayleigh, Weibull or Exponential curve. However, in many software testing environments it is difficult to describe the testing-effort function by the above three consumption curves. In this paper, we thus will show that a logistic testing-effort function can be expressed as a software development/test effort curve. Experiments have been performed based on three real.

Test debug data sets. The results show that the SRGMs with a logistic testing-effort function can estimate the number of initial faults better than previous approaches.
2. SOFTWARE RELIABILITY GROWTH MODEL

Log logistic growth model and logistic growth model have been shown to be very useful in fitting software failure data.

2.1 Logistic Testing-effort Function

The mean value function \( m(t) \) of the process of an NHPP are given by

\[
m(t) = \frac{1 - e^{-at}}{1 + e^{-at}}
\]

Since actual testing-effort data express various expenditure patterns, sometimes the testing-effort expenditures are difficult to be described by only a Exponential or Rayleigh curve. Although the [8] Weibull-type curve can fit the data well under the general software development environment, it will have an apparent peak phenomenon when the shape parameter \( m > 3 \). Therefore, we try to use a logistic testing-effort function which was first presented by Parr [9] instead of the Weibull type testing-effort consumption function as the testing-effort function to describe the test effort patterns during the software development process. This obtained function differs from the Weibull-type function described in the above subsection and was used to derive the form of the resource consumption curve of a project over its life cycle [9]. The logistic testing-effort function has the following form:

The cumulative testing effort consumption in time \( (0, t] \) is

\[
w(t) = \frac{N}{1 + Ae^{-at}}
\]

and the current testing effort consumption

\[
w(t) = \frac{dW(t)}{dt} = \frac{N\alpha e^{-at}}{(1 + Ae^{-at})^2}
\]

where \( N \) is the total amount of testing effort to be eventually consumed, \( a \) is the consumption rate of testing-effort expenditures, and \( A \) is a constant. Therefore, we can see that \( w(t) \) is a smooth bell-shaped function. The testing effort \( w(t) \) reaches its maximum value at time.

\[
t_{max} = \frac{1}{\alpha} \ln A
\]

Compared with the Weibull-type testing-effort function in the starting point, the value of logistic testing-effort function \( W(0) \) is non-zero. The divergence between the Weibull-type curve and \( W(t) \) is concentrated in the earlier stages of software development where progress is often least visible and formal accounting procedures for recording the amount of testing effort applied may not have been instituted. It is possible for us to judge between these models using some statistical test of their relative ability to fit actual failure data, such as adjusting the origin and scales linearly [10].

2.2 LLM with Imperfect Software Debugging Model

The inflection S-shaped NHPP SRGM is known as one of the flexible SRGMs that can depict both exponential and S-shaped growth curves depending upon the parameter values [11]. The model has been shown to be useful in fitting software failure data. Ohba proposed that the fault removal rate increases with time and assumed the presences of two types of errors in the software. Later, [3] modified the inflection S-shaped model and incorporated the TEF in an NHPP model. Therefore, we show how to incorporate Log-Logistic TEF into inflection S-shaped NHPP model.

The extended inflection S-shaped SRGM with Log-Logistic TEF is formulated on the following assumptions [3], [4] and [11]:

1. The software system is subject to failures at random times caused by errors remaining in the system.
2. Error removal phenomenon in software testing is modeled by NHPP.
3. The mean number of errors detected in the time interval \( (t, t + At) \) by the current testing-effort expenditures are proportional to the mean number of detectable errors in the software.
4. The proportionality increases linearly with each additional error removal.
5. Testing-effort expenditures are described by the Log-Logistic TEF [12].
6. Each time a failure occurs, the error causing that failure is immediately removed and no new errors are introduced [13].
7. Errors present in the software are of two types: mutually independent and mutually dependent.

The mutually independent errors lie on different execution paths, and mutually dependent errors lie on the same execution path. Thus, the second type of errors is detectable if and only if errors of the first type have been removed. According to these assumptions, if the error detection rate with respect to current testing-effort expenditures is proportional to the number of detectable errors in the software and the proportionality increases linearly with each additional error removal, [14],[15] we obtain the following differential equation:
\[ \frac{dm(t)}{dt} \times \frac{1}{w(t)} = \varnothing(t)(n(t) - m(t)), \]

where \( \varnothing(t) = \beta \left[ r + (1 - r) \frac{m(t)}{n(t)} \right] \)

\( r(>0) \) is the inflection rate and represents the proportion of independent errors present in the Software, \( m(t) \) be the MVF of the expected number of errors detected in time \((0, t]\), \( w(t) \) is the current testing-effort expenditure at time \( t \), \( a \) is the expected number of errors in the system, and \( b \) is the error detection rate per unit testing-effort at time \( t \).

Solving equation (1) with the initial condition that, at \( t = 0, w(t) = 0, m(t) = 0 \), we obtain the MVF

\[ m(t) = \frac{\alpha(b)^\delta [1 - e^{\beta w(t)}]}{(1 + (\beta)^\delta) + [(1 - r) / r] e^{\beta(1 - \delta) w(t)}} \]

The failure intensity at testing time \( t \) testing-effort is given by

\[ \lambda(t) = \frac{\alpha \delta \beta w(t) e^{\beta(1 - \delta) w(t)}}{(1 - \delta) [r(1 - \delta(b)^\delta) + e^{\beta(1 - \delta) w(t)}]} \]

3. COMPARISON OF PREDICTIVE CAPABILITY

The parameters of the SRGM are estimated based upon the data given below. Maximum Likelihood estimation (MLE) and Least Square estimation (LSE) techniques are used to estimate the model parameters (Musa et al., 1987; Musa, 1999; Lyu, 1996; Ahmad et al., 2008; 2010).

In order to compare predictive capability of logistic growth model and log logistic model, experiments on two actual software failure data are performed. The description of the data Sets is given in Table I[16],[17].

<table>
<thead>
<tr>
<th>Data Set</th>
<th>References</th>
<th>Errors Removed</th>
<th>Observation Period</th>
<th>Software Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS1</td>
<td>Ohba (1984)</td>
<td>328, after 3.5 years: 188</td>
<td>19 weeks</td>
<td>PL/1 application software, Execution Time: 47.65CPU hours, Size: 1317000 line of code</td>
</tr>
<tr>
<td>DS2</td>
<td>Musa et al. (1987)</td>
<td>136, after along time of testing: 338</td>
<td>21 weeks</td>
<td>Rome Air Development Center Project, Execution Time: 25.3 CPU hours, Size: 217000 line of code</td>
</tr>
</tbody>
</table>

These data sets are used to estimate the parameters of the SRGM, and the comparison results are shown in Table II.

<table>
<thead>
<tr>
<th>Model</th>
<th>( a )</th>
<th>( r )</th>
<th>( b )</th>
<th>AE (%)</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic model with Imperfect debugging</td>
<td>289.54</td>
<td>0.0252</td>
<td>9.68</td>
<td>135.87</td>
<td></td>
</tr>
<tr>
<td>Log logistic model with Imperfect debugging</td>
<td>256.88</td>
<td>0.27</td>
<td>0.0263</td>
<td>6.23</td>
<td>67.87</td>
</tr>
</tbody>
</table>

We compute the relative error in prediction of logistic growth model and log logistic growth model for this data set. Figures 1 and 2 show the relative error plotted against the percentage of data used (that is, \( t_e / t_q \)). Figures and table reveal that the log logistic growth model predicts the future behavior well as compare to logistic growth model [20].

DS 1: Table II lists the comparisons of logistic growth model and log logistic growth model SRGMs which reveal that the log logistic growth model has better performance[19].

DS 2: Table III shows the comparisons of logistic model and log logistic model with different SRGMs which reveal that the log logistic model has better performance for this data set.
<table>
<thead>
<tr>
<th>Model</th>
<th>a</th>
<th>r</th>
<th>AE (%)</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic model with Imperfect debugging</td>
<td>654.08</td>
<td>0.0427</td>
<td>10.06</td>
<td>118.59</td>
</tr>
<tr>
<td>Log logistic model with Imperfect debugging</td>
<td>685.63</td>
<td>0.37</td>
<td>9.54</td>
<td>87.69</td>
</tr>
</tbody>
</table>

The relative error in prediction is calculated for logistic growth model and log logistic and the results are shown graphically in Figures 3 and 4. Finally, from the Figures 3 to 4 and Table III, it can be concluded that the log logistic model gets reasonable prediction as compare to logistic model.

**4. CONCLUSION**

In this paper we discuss exponential type and log logistic type SRGM with imperfect debugging testing-effort. We analyzed the Predictive capability of logistic and log logistic growth models for the actual data applications. We then compared its predictive capability graphically. The findings reveal that log logistic type SRGM has better prediction capability as compare to logistic type SRGM.

**ACKNOWLEDGMENTS**

The authors are pleased to thank the referee for their helpful comments and suggestions, which led to improvements in this paper.

**REFERENCES**


