Measuring Software Reliability and Release Time Using SRGM Tool

B. Anni Princy¹, Dr. S. Sridhar²

¹Research Scholar, Sathyabama University, Chennai,
²Research Supervisor, Sathyabama University, Chennai,
Email-id: ¹shiyaprincy@yahoo.com, ²drssridhar@yahoo.com

ABSTRACT

Software systems are used in numerous significant applications. They all need high quality, reliability and safety. In our paper we present a software reliability growth model for estimating the software error, failure rate and thus calculating the reliability. To boost up the detected error and correction a great attempt is required which alternatively increase the reliability. For that we are introducing Gompertz testing effort function and analysis is done on optimal release time. It describes the number of persons involved, number of test cases and calendar time. Here we use Yamada delayed S-shaped model in order to isolate the software error. Experiments are performed on real datasets.

Keywords- Software Reliability Growth Model(SRGM), Yamada Delayed S-Shape, Optimal Release Time, Gompertz Testing Effort Function

INTRODUCTION

The complexity and size of a computer system have grown significantly for the past two decades. Usually, most researches focused only on the design, improvement and reliability analysis of the hardware to reach the goal of high-concert computer systems. But now, the increasing trend of software criticality has generated more researches into the field of high-superiority software progress. In extremely composite contemporary software systems, reliability is the most important factor since it quantifies software failures during the process of software development and software quality control. Software reliability is the likelihood that given software will be operation without malfunction in a given surroundings for the period of a particular phase of time [8-10]. Software reliability engineering is centered on a key attribute, software
reliability, which is defined as the probability of failure-free software operation for a specified period of time in a specified environment [11].

To evaluate modular software quantitatively and expect the reliability of each of the modules for the period of module testing, software reliability growth models (SRGM) are used. Several SRGM’s, which narrate the number of failures (fault identified) and the Execution time (CPU time/Calendar time) have been discussed in the literature [12, 13, 14]. 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 status and SRE technology quantitatively [17].Reliability is defined as the working situation of the software over certain time period of time in a given ecological conditions. Large numbers of papers are presented in this context. Testing effort is defined as effort needed to detect and correct the errors during the testing. Testing-effort can be calculated as person/ month, CPU hours and number of test cases and so on.

In general the software testing consumes a testing-effort during the testing phase [15, 16]. The simplest appearance of a software reliability growth model is an exponential one. However, S-shaped software reliability growth is more often observed than the exponential one in actual projects. The Gompertz curve model gave good approximations to collective number of software faults observed in testing software for a switching system. However, the Gompertz equation, which is a differential equation, has to be estimated as a difference equation to obtain a weakening equation. This means that even if the model itself is appropriate for the data, we still cannot get the right parameters [18]. To our knowledge, however, SRGMs have not been used with regard to fault endow predictions. Several authors claim that SRGMs might not be applicable in practice at all as the model’s assumptions are often despoiled [19-20].

As a measure of time, calendar time is used, i.e. the week when the fault is reported. It has been shown that SRGMs perform well based on calendar time [21]. These SRGMs can be further categorized into S-shaped and concave models respectively, which relates to the common attitude of their curves. In particular, the curves of concave models bend downwards, while curves of S-shaped models first congregate and become concave later [22]. The concave models and the linear model perform equally well, i.e. with no significant difference. Nevertheless, SRGMs require a certain mathematical background and hence, practitioners might be reluctant to use such models [13]. Different fault detection methods are interesting because they supply valuable data that forms the basis for evaluating software reliability Therefore, the main concern is centered around software faults, their effect on the system and the residual number of faults, system failures, the way of detecting failures, time between failures and failure rates, as well as the confidence in the performed estimates [23].

RELATED WORKS

The use of open source software was suitable more and more leading and it was important that the reliability of that software was evaluated. Even though a lot of researchers have tried to establish the failure pattern of
different packages a deterministic model for evaluating reliability was not yet developed. Shelbi Joseph et al. [1] presented work details a simplified model for evaluating the reliability of the open source software based on the available failure data. The methodology involves identifying a fixed number of packages at the start of the time and defining the failure rate based on the failure data for these preset number of packages. The defined function of the failure rate was used to arrive at the reliability model. The reliability values obtained using the developed model was also compared with the exact reliability values.

Software Reliability Growth Models (SRGMs) were very important for estimating and predicting software reliability. Several combinational methods of SRGMs have been proposed by Y Vamsidhar et al. [2] to improve the reliability estimation and prediction accuracy. The AdaBoosting (Adaptive Boosting) algorithm was one of the most popular machine learning algorithms. An AdaBoosting based approach for obtaining a dynamic weighted linear Combinational Model (ACM) was already proposed. The key idea of that approach was that they selected several SRGMs as the weak predictors and use AdaBoosting algorithm to determine the weights of these models for obtaining the final linear combinational model.

RajPal Garg et al. [3] presented a computational methodology based on matrix operations for a computer based solution to the problem of performance analysis of software reliability models (SRMs). A set of seven comparison criteria have been formulated to rank various non-homogenous Poisson process software reliability models proposed during the past 30 years to estimate software reliability measures such as the number of remaining faults, software failure rate, and software reliability. Selection of optimal SRM for use in a particular case has been an area of interest for researchers in the field of software reliability. Tools and techniques for software reliability model selection found in the literature cannot be used with high level of confidence as they used a limited number of model selection criteria. A real data set of middle size software project from published papers has been used for demonstration of matrix method. The result of that study would be a ranking of SRMs based on the Permanent value of the criteria matrix formed for each model based on the comparison criteria. The software reliability model with highest value of the Permanent was ranked at number – 1 and so on.

Software reliability was one of the important factors of software quality. Before software delivered in to market it was thoroughly checked and errors were removed. Every software industry wants to develop software that should be error free. Software reliability growth models were helping the software industries to develop software which was error free and reliable. Shaik.Mohammad Rafi et al. [4] performed an analysis based on incorporating the logistic-exponential testing-effort in to NHPP Software reliability growth model and also observed its release policy. Experiments are performed on the real datasets.

In software development life cycle, software testing was one of the most important tasks; and in the testing, software reliably was very important aspect for any category of software systems. A number of testing-effort functions for software reliability growth model based on non-homogeneous Poisson process (NHPP) have been proposed in the past. Although these models were quite helpful for software developers and have been
widely accepted and applied in the industries and research centers, they still need to put more testing-effort functions into software reliability growth model for accuracy on estimate of the parameters. S. M. K. Quadri et al. [5] considered the case where the time dependent behaviors of testing-effort expenditures were described by New Modified Weibull Distribution (NMWD). Software Reliability Growth Models (SRGM) based on the NHPP were developed which incorporates the (NMWD) testing-effort expenditure during the software testing phase. It was assumed that the error detection rate to the amount of testing-effort spent during the testing phase was proportional to the current error content. Model parameters were estimated by Least Square and Maximum Likelihood estimation techniques, and software measures were investigated through numerical experiments on real data from various software projects. The evaluation results were analyzed and compared with other existing models to show that the proposed SRGM with (NMWD) testing-effort has a fairly better faults prediction capability and it depicts the real-life situation more faithfully. That model could be applied to a wide range of software system. In addition, the optimal release policy for that model, based on reliability criterion was discussed.

The application of computer system has now crossed many different fields. Software has become an essential part of many industrial, military and even commercial systems. Both in microcomputers and supercomputers, they might find programs containing millions of lines of code. That has, together with the application of software in many safety critical systems, led to the fact that software reliability was now an important research area. Khalaf Khatatneh et al. [6] explored a model that could be used for software reliability prediction. The proposed model was implemented using the fuzzy logic technique and has been applied on a custom set of test data. The model was characterized as a growth reliability model. That model focused on a particular dataset behavior in predicting reliability. Focusing on a particular dataset behavior was performed to develop an accurate model since the recent work focused on developing a model which could be more accurate.

Software reliability was one of the most important characteristics of software quality. Its measurement and management technologies employed during the software life-cycle were essential for producing and maintaining quality/reliable software systems. Herein, D. R. Prince Williams [7] discussed a modified approach to calculating the delivery cost of a software product, when warranty was to be provided, with an imperfect debugging phenomenon. Unlike existing cost models, there the strategy was to consider maintenance support given to the customer with an imperfect debugging phenomenon. The optimal release time could be calculated for various reliability levels by minimizing the cost. The delivery cost, reliability of the software system, and the optimal release time were calculated by using an imperfect debugging software reliability growth model. Numerical illustration supports the optimal release policies.
SOFTWARE RELIABILITY MODEL

1. Software Reliability Growth Models (SRGMs)

Software reliability growth models with a model are formulated by a Non-Homogeneous Poisson Process (NHPP). By using the least square estimation or maximum likelihood estimation method SRGM parameters are computed and thus by using and using actual software failure data, numerical results are obtained. An ideal software reliability growth model must offer consistently accurate reliability estimation. It is observed that there is no such model which can attain accurate results for different cases. The reason behind that is in many models it is shown to perform well with one failure data-set, but bad with the other data-set. In our proposed method we are anticipating software testing effort function with software reliability growth model. The detailed description is given below.

2. PROPOSED METHOD

In general, software testing effort can be defined as the amount of effort spends during the software testing. Testing-effort can be described by following curves. Plenty of curves are proposed in literature to express the testing-effort. Testing effort is described as amount of testing expenditure is spend during the testing.

A) Discrete exponential curve: let W(n) denote the expected cumulative number of faults detected up to nth testing-period. Discrete analog exponential curve as ‘b’ represents rate at which testing effort is consumed, ‘a’ represents the initial total test effort before the testing begins.

\[ W(n+1) - W(n) = d \times b \times (a - W(n)) \]  

Cumulative testing effort can described in (0, t):

\[ W(t) = \alpha (1 - e^{-\beta t}) \]  

Where \( \alpha \) the total is amount of testing expenditure and \( \beta \) is the consumption rate of the testing effort.

B) Gompertz Curve: generally the testing-effort consumption is slow at the beginning of the test phase; all the members of the testing team should be familiar with the testing process and its internal details. One all the team members are familiar with testing consumption of testing effort increases. This unusual nature gives the testing-effort to derive the S shaped. The Gompertz Cumulative Testing-effort in (0, t] is given by

\[ W(t) = \alpha e^{-\beta e^{-\gamma t}} \]  

C) Log-Logistic TEF : The log-logistic distribution is the probability distribution of a random
variable whose logarithm has a logistic distribution. It is similar in shape to the log-normal distribution but has heavier tails. Logistic Curve model is one of the S-shaped growth models. Discrete analog Logistic TEF is given by

\[ W(t) = \frac{\alpha}{1 + \left(\frac{t}{\lambda}\right)^{\beta}} \quad (4) \]

2.1 YAMADA DELAYED s-SHAPED WITH GTEF

In Yamada delayed S-shaped model test the software not only of error detection but it isolates the error also. Thus the cumulative error follows the s-shape curve. The extended Yamada s-shaped model is given by:

\[ \frac{dm_d(t)}{d(t)} \times \frac{1}{w(t)} = r_1 \times [a - m_d(t)] \quad (5) \]

And \[ \frac{dm_r(t)}{d(t)} \times \frac{1}{w(t)} = r_2 \times [a - m_r(t)] \quad (6) \]

Suppose \( r_2 \neq r_1 \) solving above equations at boundary condition \( m_d(t) = 0 \),

\[ m_d(t) = a \times \left(1 - e^{-r_1 \times W^*(t)}\right) \]

And \[ m_r(t) = a \times \left[1 - \frac{r_1 \times e^{-r_2 \times W^*(t)} - r_2 \times e^{-r_1 \times W^*(t)}}{r_1 - r_2}\right] \quad (7) \]

Assuming \( r_2 \approx r_1 \approx r \) and applying L Hospital's rule the delayed S-shape model with TEF is given by

\[ m(t) \equiv m_r(t) = a \times \left(1 - (1 + r \times W^*(t)) \times e^{-r \times W^*(t)}\right) \quad (8) \]

2.2 SRGM WITH GOMPertz TESTING-EFFORT FUNCTION

In our proposed method we are using testing effort modeled by a Gompertz TEF. When each time a failure occurs the fault that cause it is removed immediately. And also no new faults are introduced. The mathematical expression of testing effort is given by

\[ \frac{dm_d(t)}{d(t)} \times \frac{1}{w(t)} = r_1 \times [a - m(t)] \quad (9) \]

After giving boundary condition \( m(0) = 0 \) and \( r(t) = r(0 < r < 1) \),

\[ m(t) = a \times (1 - e^{-r(t-W(t))^r}) \quad (10) \]

Substituting \( W(t) \) value from (3)

\[ m(t) = a \times (1 - e^{-(aw^r - \alpha - \beta)} - \alpha - \beta) \quad (11) \]
The number of faults remaining in system is 
\[ a - m(t) = m_{remaining}(t) = a e^{-(W(t)-W(0))} \]

### 2.3 RELEASE TIME on RELIABILITY

Release time is the time for the software to be delivered out. The conditional reliability function after the last failure at time \( t \) is given by:
\[ R(t + \Delta t / t) = \exp\left(-\left[m(t + \Delta t / t) - m(t)\right]\right) \]
\[ = \exp(-m(\Delta t) \times \exp(-r \times W^*(t))) \quad (13) \]

Taking logarithm on both sides
\[ \ln R = -m(\Delta t) \times (-r \times W^*(t)) \quad (14) \]

Thus the reliability is given by
\[ R(T) = m(t) / a \quad (16) \]

### 2.4 COST AND EFFICIENCY CALCULATION USING RELEASE TIME

Increase in the time to test the software increases the complexity of software. It is always observed that allotted time for testing of software can exceed its required schedule time. To speedup the testing process automated testing tools are used. It does not only speedup the testing process but it increases the efficiency of the testing by certain extend. The total cost of software is given as:
\[ C_2(T) = C_0(T) + C_1(1 + P)m(T) + C_2\left[m(T_{te}) - (1 + P)m(T)\right] + C_3\left[\int_0^T w(t)dt\right] \quad (17) \]

Where \( P \) is described as fractions of extra errors found during the software testing phase and is the number of additional faults during the testing time. \( C_0(t) \) is cost of adopting a new automated testing tools into testing phase. \( P \) is directly proportional to cost as \( P \) increases cost also increases. The total cost of the software is summation of cost of correcting the errors before and after the release of software. \( C_1 \) cost of correcting the error during the testing, \( C_2 \) cost of correcting an error during operation and \( C_3 \) cost of testing per unit testing expenditure. The release time is given by:
\[ T_0 = -\frac{\ln\left(-\frac{r\alpha + \ln\left(-\frac{C_3}{ar(-C_2 + C_1)}\right)^a}{b}\right)}{c} \quad (18) \]
RESULT AND DISCUSSIONS

An SRGM is generally a tool that can be used to evaluate the quantitative nature of any software, develop test status, schedule status, and monitor the changes in reliability performance. Software reliability assessment and prediction is important to evaluate the performance of software system. In our paper we have designed an SRGM model with Gompertz testing effort function. The proposed method is implemented in the working platform of JAVA and the results obtained are evaluated to check the performance of our proposed methodology.

The table 1 given below shows the time and the no of failures evolved while testing the software. The corresponding graphical representation of the respective values is shown in the fig 1,

Table1: Number of failures at various times

<table>
<thead>
<tr>
<th>Time</th>
<th>No of failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.097</td>
<td>2</td>
</tr>
<tr>
<td>0.14</td>
<td>3</td>
</tr>
<tr>
<td>0.187</td>
<td>3</td>
</tr>
<tr>
<td>0.329</td>
<td>2</td>
</tr>
<tr>
<td>0.348</td>
<td>2</td>
</tr>
<tr>
<td>0.38</td>
<td>2</td>
</tr>
</tbody>
</table>
From the above graphical representation the corresponding failures of testing the software at respective time can be evaluated.

The table 2 given below shows the values of reliability and cost of the particular software obtained with varying time.

**Table 2:** Reliability and Cost estimate for corresponding time.

<table>
<thead>
<tr>
<th>Time</th>
<th>Reliability</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.097</td>
<td>0.005927</td>
<td>43223.13</td>
</tr>
<tr>
<td>0.14</td>
<td>0.005929</td>
<td>43222.28</td>
</tr>
<tr>
<td>0.187</td>
<td>0.005931</td>
<td>43221.42</td>
</tr>
<tr>
<td>0.329</td>
<td>0.005933</td>
<td>43220.57</td>
</tr>
<tr>
<td>0.348</td>
<td>0.005935</td>
<td>43219.72</td>
</tr>
<tr>
<td>0.38</td>
<td>0.005938</td>
<td>43218.86</td>
</tr>
</tbody>
</table>

The graphical representation for the reliability and time as well as the cost and time are given as shown in the fig 2 and fig 3 respectively.
**CONCLUSION**

In this paper we have developed a general approach in deriving more general models based on simple assumptions, constant with the basic software reliability growth modeling with Gompertz testing effort function. Incorporating the dynamics of testing time of the software and the testing coverage has allowed us the model to be a two dimensional framework. The reliability and cost values are measured and the performances of the proposed methods are evaluated. The proposed models are validated on real data sets and analyses are done using goodness of fit criterion. The results prove that our proposed method of
software reliability growth modeling is more effective and better model when compared to other related works.

REFERENCES