Software Reliability Growth Models with Exponentiated-gompertz Testing Effort and Release Time Determination

Ahmad Raad Raheem*
Department of Computer Science and Engineering, Acharya Nagarjuna Engineering, A.P, India
E-mail: ahmedraadr@yahoo.com
*Corresponding Author

Shaheda Akthar
Department of Computer Science, Government College for Women(A), Guntur, A.P, India
E-mail: shahedaakthar76@gmail.com

Received: 28 June 2021; Revised: 04 December 2021; Accepted: 11 April 2022; Published: 08 February 2023

Abstract: Quality is a consequential factor for the software product. During the software development at most care was taken at each step for the quality product. Development process generally embedded with several qualitative and quantitative techniques. The characteristics of final software product should reach all the standards. Reliability is a paramount element which quantifications the probability that a software product could able to work afore it authentically fails to perform its intended functionality. Software testing is paramount phase where gargantuan resources were consumed. Over around fifty percent of cost was consumed during this testing phase, that is why testing was performed in disciplined environment. Software product release time is considered to be crucial subject at which the software product testing was stopped and it could be release into market, such that the software product should have quality and reliability. In this paper we have investigated the concept of software testing effort dependent software reliability growth models by considering the exponentiated-gompertz function as testing effort function to determine the release time of the software. Thus, constructed testing effort dependent models was computed on three authentic time datasets. Parameter estimation is done through least square estimation and metrics like Mean square Error (MSE) and Absolute Error (AE) are utilized for model comparison. The proposed testing effort dependent model performance was better than the rest of the models.

Index Terms: Software Testing, Testing Effort, Software Reliability, Software Cost, Mean Square Error, Absolute Error.

1. Introduction

The development and modernization of industries has increased the greater extent of modern devices, whether the hospitals, business companies, telecommunication and aeronautics. These modern gadgets and computing devises need always software thus software should have more quality and reliability. Software industries struggling from past few decades to prepare and design a quality and reliable software as the development process its self is fuzzy and complex in nature. Now a days industries are adapting new techniques and metrics to achieve the intended quality and reliability. Software reliability was defined as probability of software product could able to work for certain period of time before it could actually fail. Usually, software development activities are amalgamated with several tasks. Software testing is considered to be the most important phase among all the phases where heavy resources were consumed. Software testing phase usually done in systematic environment by considering all resource and time aspects. Past few years several authors have proposed different types of Software reliability growth models under Nonhomogeneous Poisson process models, where each model have its own pros and cons. [1] Designed a Nonhomogeneous Poisson process stochastic model where failure intensity is decreases exponentially with time. [2] they assumed that the model follows the failure detection rate which varies with time. [3] has proposed an S shaped model inflection model by incorporating inflection parameter in their model. Usually most of the software reliability growth models comes under either
exponential or S-shaped family depending on the kind of assumption they made during the model design. [4-6] Some authors have proposed software reliability models based on the relation between failure intensity and its test coverage, they grouped these models under test coverage models. [7-9] In recent years, there has been considerable interest shown in release time determination of the software; several models were proposed under this assumption. [10-18] Several studies were carried out to investigate the inclusion of testing effort into the software reliability growth models. Testing effort is defined as the number of test cases used for testing, number of people allotted for testing and fault determination time. Much of the work was carried out by considering the various testing effort related functions in Software reliability growth models, however, there were still some models unable to fit the data in all circumstances. So, in this study, we have proposed a new [19, 20] exponentiated-Gompertz distribution as testing effort into the software reliability growth model. The reason for using proposed testing effort function is, basically, failure datasets have either exponential or S-shaped distribution characteristic. In this paper, we have investigated functional characteristic of software reliability growth model by considering Exponentiated-Gompertz testing effort function.

These Nonhomogeneous Poisson process software reliability growth models usually have the stochastic in nature [21]. These models have failure intensity function which was represented by \( \lambda(t) \). \( N(t) \) represents the number of failures in the software at a time \( t \) and \( m(t) \) represents the mean or expectation. This mean value is computed through the following equation. The relation between mean and intensity of failure data was given by \( E[N(t)]=m(t) \) and the relation between mean and failure intensity is given by

\[
m(t) = \int_0^t \lambda(t)dt 
\]

(1)

\( N(t) \) have the probability mass function which was given by

\[
Pr\{N(t) = n\} = \frac{(m(t))^n e^{-m(t)}}{n!} \quad n = 0,1,2,\ldots,\infty
\]

(2)

This paper is organized as follows; Section 2 describes the various testing efforts related to Nonhomogeneous Poisson process models. Section 3 briefly overviews Software reliability growth models and their properties. Section 4 describes the various failure datasets were used for study. Section 5 provides the detailed information on performance metrics used for evaluation of models. Section 6 describes the total review on results and discussion and section 7 conclusion.

2. Related Works

The Poisson model [22] decreases the fault rate in geometric progression. Exponential, normal, gamma and Weibull functions are also used to design the SRGM. The SRGMS are abstractly unsuitable [23] for various proprietary data sets cross study comparisons and leads to an inadequate usage of statistical testing results. For NHPP, the Exponential Weibull testing effort function (TEF) [24] is used for designing the inflection S-shaped SRGMs. It is found to be flexible with imperfect debugging. For Noisy input, the analysis of sensitivity [25] is required to measure the conclusion stability in terms of noise levels concerned. The genetic programming based on symbolic regression [26] is found to be efficient for SRGM design. A hierarchical quality model [27, 28] based SRGM is automatically calculates the metric values and their correlation with various quality profiles. The distance-based approach (DBA) [29] based SRGM is evaluated for selecting the optimal parameters and yields the ranking. It identifies the importance of criteria for the application. The Japanese software development system utilizes the Gompertz curve [30] for residual faults estimation. In SRGM, multiple change points are crucial for detection of environment changes [31]. These are known for efficiently handling both the imperfect and ideal debugging conditions. In SRGM, application of Queueing modes is used for describing the fault detection rate and correction procedure. For this, the extension of infinite server queuing models [32, 33] is used based on multiple change points. The Component Based Software Development (CBSD) [34, 35] is considered as building blocks for SRGM. The CBSD primarily focuses on selection of appropriate user requirements. Some of the soft errors can be minimized by using minimum redundancy concept of critical data [36, 37]. In [38] Anjum et al., presented a ranking based weighted criteria for reliability assessment where the failure data sets are ranked accordingly.

3. Methodology

The present paper proposes a novel approach for quality assessment for software growth model. Contribution of Paper discussed in section A and Motivation behind this Work is discussed in section B.

3.1. Contribution of Paper

Non-Homogeneous Poisson Process Software Reliability Growth Models with Generalized Gompertz TEF

Software reliability growth model with Generalized Gompertz testing effort was given by the following assumptions [11-13, 14-17].

Volume 15 (2023), Issue 1 82
The fault removal process follows the Nonhomogeneous Poisson process models (NHPP).

The software system is subjected to failure at random time caused by fault remaining in the system.

The mean time number of faults detected in the time interval \((t, t+\Delta t)\) by the current test effort is proportional for the mean number of remaining faults in the system.

The proportionality is constant over the time.

Consumption curve of testing effort is modeled by a Generalized Gompertz TEF.

Each time a failure occurs, the fault that caused it is immediately removed and no new faults are introduced.

We can describe the mathematical expression of a testing-effort based on following

\[
\frac{dm(t)}{dt} \times \frac{1}{\omega(t)} = b(a - m(t)) \quad (3)
\]

In the equation (3) parameters ‘a’ represents total number of faults before the testing begins and ‘b’ represents the failure detection rate. Solving the equation (3) assuming \(m(0)=0\) at \(t=0\) we get the required mean value function as

\[
m(t) = a(1 - e^{-b(W(t)-W(0)))}) \quad (4)
\]

From the equation (5) Generalized Gompertz testing function produces the value zero when \(t=0\). Now substitute the equation (5) into the equation (4) to the required software reliability growth model with proposed testing effort function.

\[
m(t) = a\left(1 - e^{-b\left(\phi\left(1-e^{-\mu(\omega-1)}\right)\right)}\right) \quad (5)
\]

From the equation (4) failure intensity is computed by

\[
\lambda(t) = ab\omega(t)e^{-W(t)} \quad (6)
\]

Now substitute the equation (7) into the equation (6) we get

\[
\lambda(t) = ab\left(\phi\left(1-e^{-\mu(\omega-1)}\right)\right)^{\delta-1} \omega e^{-\mu\omega\Delta t+\mu\Delta t} e^{-\left(\phi\left(1-e^{-\mu(\omega-1)}\right)\right)} \quad (7)
\]

3.2. Motivation behind this Work

Various testing efforts used related to non-homogeneous poisson process software reliability growth models.

Since in the era of software reliability growth models sundry authors have proposed different software testing effort functions. Conventionally testing effort plays a consequential role in software reliability growth model, and can be expressed in terms of number of testing persons, number of test cases and time required to find the number of faults during testing.

**Exponential, Rayleigh curve and Weibull Curve [11,13,17]: These testing effort functions were proposed by Yamada in his research papers.**

\[
W_e(t) = (1 - e^{-rt}) \quad (8)
\]

\[
W_R(t) = (1 - e^{-rt^2}) \quad (9)
\]

\[
W_{we}(t) = (1 - e^{-rt^m}) \quad (10)
\]

Equation (3), (4) and (5) represents the testing effort Curves cognate to exponential Rayleigh and Weibull. \(W_e(t)\), \(W_R(t)\), and \(W_{we}(t)\) represents the cumulative testing efforts of exponential, Rayleigh and Weibull curve. Exponential curve describes the process deteriorating monotonically. Whereas Rayleigh curve shows the different nature as compared to exponential unlike exponential curve it first increases and after reaches to certain level it declines. Weibull curve shows the different variations depending on the value of \(m\), and curve reaches to maximum at \(m=3\).

**Logistic curve [16]:**

Logistic curve has many advantages in several applications, because the characteristic of authentic time datasets has very near sodality among themselves.

\[
W_L(t) = \frac{A}{(1+Be^{-rt})} \quad (11)
\]
\( W_c(t) \) represents the logistic curve cumulative testing effort. In the equation (11) \( \beta \) represents the inflection parameter.

**Log-Logistic Curve [21]:**

This curve was used by the authors to capture the increasing and decreasing phenomenon in failure intensity function. \( W_{LL}(t) \) represents the cumulative log-logistic testing effort function.

\[
W_{LL}(t) = N \left( \frac{\alpha}{1 + \alpha e^{-\lambda t}} \right)^k
\]

**Exponentiated-Gompertz(Generalized Gompertz) Distribution[19,20]:**

This distribution was proposed by A. El-Gohary, Ahmad Alshamrani, Adel Naif Al-Otaibi. Main advantage of this testing effort function is shows various failure intensity increase, decreasing, constant and bath tub depending on the shape parameter. The Generalized Gompertz cumulative testing effort is given by

\[
W_{EG}(t) = \phi \left[ (1 - e^{-\mu(t)})^\delta \right] \quad \mu > 0, \eta > 0 \text{ and } \delta > 0
\]

\( W_{EG}(t) \) represents the cumulative testing effort function, \( \delta \) represents the shape parameter.

\[
W_{EG}(t) = \int_0^t \omega(t)dt
\]

\( \omega(t) \) represents current testing effort expenditure which was represented by the equation (15)

\[
\omega(t) = \phi \left[ (1 - e^{-\mu(t)})^\delta \right] \delta \mu \eta e^{-\mu(t)} + \mu + \eta
\]

This generalized Gompertz exhibits various special cases in the forms of distributions depending on the values of parameters.

- Generalized exponential distribution with parameter \( (\mu, \delta) \) can obtained by making parameter \( \eta \) to approaches to zero.
- Generalized Gompertz with parameters \( (\mu, \eta) \) can be obtained by assigning the zero to parameter \( \delta \).
- One parameter \( \mu \) exponential model can be obtained by making parameter \( \eta \) to approaches to zero and assign the parameter \( \delta \) to 1.

**Failure Datasets used in this Research**

- **DS1[3]:** This dataset was drawn from the work carried out by Ohba 1984, where data belongs to the PL/1 database application software consisting of approximately 1,317,000 lines of code. A total of nineteen and 47.65 CPU hours were spend to capture 328 software errors are removed. The test was continued further in order to recover further errors, and they have recovered round 358 errors.
- **DS2[15]:** The second dataset was cited from the research papers published by Musa 1987 and Musa 1999. Name of the system was T1 and called Rome air development project. The size of the software approximately 21700 lines of code around 21 weeks and 25.3 CPU hours was spent to discover around 136 faults were reported. A total of 93 CPU hours spends during software testing.
- **DS3:** Third dataset was belonging to the research paper published day Tomha [39]. In this cumulative number of software bugs discovered 86 and 22 days spend during testing.

4. **Model Performance Metrics and Analysis**

4.1. **Parameter Estimation Methods [40,14]**

Here we used least square parameter estimation to estimate the \( \phi, \delta, \eta, \) and \( \mu \) parameters of the model. These parameters are estimated for the dataset which is in the form \((x_i, W_i) < x_1, x_2, x_3, \ldots x_n > \) and \(< W_1, W_2, W_3, \ldots W_n >\) and \((m_i) < m_1, m_2, m_3, \ldots m_n >\); here \( x_i \) represents the time of testing, \( W_i \) represents the cumulative testing effort, \( m_i \) represents the cumulative errors in the given datasets and ‘n’ represents the number of error terms in the datasets. Following is the expression in which we can derive the residual sum of square

\[
S(\phi, \mu, \eta, \delta) = \sum_{i=1}^{n} (W_i - W(t_i))^2
\]

Equation (16) can be expressed in another form by substituting the equation (8).
Software Reliability Growth Models with Exponentiated-gompertz Testing Effort and Release Time Determination

\[ S(\phi, \mu, \eta, \delta) = \sum_{i=1}^{n} \left( W_i - \left( \phi \left[ \left( 1 - e^{-\mu \left( \eta (e^{\tau_i} - 1) \right)} \right) \right] \right)^2 \]  

(17)

To estimate the parameters, we simply partially differentiate the equation (17) with respect to each parameter and equating the resultant equation to zero, which was shown in the equation (18).

\[ \frac{\partial S(\phi, \mu, \eta, \delta)}{\partial \phi} = 0 \]  

(18)

In this paper we used python-based program for scipy library with curve fit function for parameter estimation.

4.2. Performance Analysis and Evaluation Metrics [16,18]

Performance and evaluation metrics are important for analysis and comparative study of various models with proposed model. The following are the metrics were used in this paper for comparative and performance measure.

- **MSE:** (mean square error): This metric is useful to measure the difference between actual and estimated values of cumulative errors.

\[ MSE = \sum_{i=1}^{n} \frac{(m(t_i) - y_i)^2}{k} \]  

(19)

Lower value of MSE indicates fewer fitting errors, and excellent goodness of fitting.

- **Prediction error:** It measure the difference between actual and estimated number of errors at any instant of time.

\[ PE_i = (m_i - y_i) \]  

(20)

Average value of prediction error known as Bias.

\[ Bias = \frac{\sum_{i=1}^{n} (m_i - y_i)}{n} \]  

(21)

Lower the value of Bias indicates better goodness of fit.

- **Variation:** The standard deviation of Bias and prediction error known as Variation. Lower the value of Variation indicates better goodness of fit.

\[ Variation = \sqrt{\frac{\sum_{i=1}^{n} (PE_i - Bias)^2}{n-1}} \]  

(22)

- **Root Mean Square Error Prediction Error:** It measures how close is our predicted value with estimated value. Lowe the value indicates the better goodness of fit.

\[ RMSPE = \sqrt{(Bias)^2 + (Variation)^2} \]  

(23)

Lower the value of RMSPE indicates better goodness of fit.

- **AE (Accuracy of estimation):** It is defined as the ratio of difference in actual total number of errors and total number of errors estimated at the end of the testing by the estimated total number of errors at the end of testing. Here \( M_a \) is the total number of errors estimated and ‘a’ is actual number of errors detected at the end of the testing.

\[ AE = \left| \frac{M_a - a}{M_a} \right| \]  

(24)

5. Experimental Results and Discussion

Figure (1) shows the fitting of testing effort to the actual dataset 1. By using least square parameter estimation was used to estimate the parameters of our proposed testing effort model and estimated model parameters are \( \phi = 61.32, \mu = 0.2585, \eta = 0.0982, and \delta = 0.907 \). Figure 1 shows the proposed testing effort curve and actual testing effort. As from the figure 1 shows that out proposed curve fits well and completely captures the real testing effort.
Figure (2) shows the fitting of testing effort to the actual dataset 2. By using least square parameter estimation was used to estimate the parameters of our proposed testing effort model and estimated model parameters are $\phi = 25.88, \mu = 0.00395, \eta = 0.0328, \text{and } \delta = 1.408$. Figure 2 shows the proposed testing effort curve and actual testing effort. As from the figure 1 shows that our proposed curve fits well and completely captures the real testing effort.

Figure (3) shows the fitting of testing effort to the actual dataset 3. By using least square parameter estimation was used to estimate the parameters of our proposed testing effort model and estimated model parameters are $\phi = 96.67, \mu = 0.2055, \eta = 0.1269, \text{and } \delta = 1.202$. Figure 3 shows the proposed testing effort curve and actual testing effort. As from the figure 1 shows that our proposed curve fits well and completely captures the real testing effort.

Table 1 describes different testing effort functions and their relative performance metrics for the dataset 1.

Figure 4 shows the relative error curve for the dataset 1, which specifies how far our proposed model follows the actual failure phenomenon.

Table 2 describes various models and their respective parameters and performance metrics like AE and MSE for
the dataset 1. The software reliability growth model with proposed testing effort has \(a=565.1\), \(b=0.0196\), \(\text{AE}=57.84\) and \(\text{MSE}=103.06\). From the Table 2 model with proposed testing effort had less AE and MSE values compared with other models.

Table 1. Performance Analysis of Proposed Testing Effort Function and other Functions for Dataset1.

<table>
<thead>
<tr>
<th>Function</th>
<th>Bias</th>
<th>Variation</th>
<th>RMS-PE</th>
<th>PE (at the ending of the testing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponentiated-Gompertz</td>
<td>0.0326</td>
<td>0.93</td>
<td>0.90</td>
<td>0.10</td>
</tr>
<tr>
<td>Exponentiated Weibull</td>
<td>0.0446</td>
<td>0.95</td>
<td>0.92</td>
<td>-0.09</td>
</tr>
<tr>
<td>Burr type X</td>
<td>0.038</td>
<td>0.95</td>
<td>0.93</td>
<td>-0.03</td>
</tr>
<tr>
<td>Yamada exponential</td>
<td>-0.39</td>
<td>1.37</td>
<td>1.42</td>
<td>1.27</td>
</tr>
<tr>
<td>Yamada Rayleigh</td>
<td>0.83</td>
<td>2.17</td>
<td>2.27</td>
<td>2.50</td>
</tr>
<tr>
<td>Yamada Weibull</td>
<td>0.03</td>
<td>0.96</td>
<td>0.93</td>
<td>-0.38</td>
</tr>
<tr>
<td>Huang logistic</td>
<td>-0.10</td>
<td>1.31</td>
<td>1.28</td>
<td>1.05</td>
</tr>
</tbody>
</table>

Table 2. Performance Analysis of Proposed Software Reliability Model and other Models for Dataset1.

<table>
<thead>
<tr>
<th>MODEL</th>
<th>(a)</th>
<th>(b)</th>
<th>AE(%)</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM with eq. proposed TEF eq.13</td>
<td>565.1</td>
<td>0.0196</td>
<td>57.84</td>
<td>103.06</td>
</tr>
<tr>
<td>SRGM with Exponentiated Weibull [14]</td>
<td>565.4</td>
<td>0.0196</td>
<td>57.93</td>
<td>113.10</td>
</tr>
<tr>
<td>SRGM with [14] Burr type X</td>
<td>565.7</td>
<td>0.0196</td>
<td>69.15</td>
<td>123.67</td>
</tr>
<tr>
<td>Yamada Exponential model with exponential curve [41]</td>
<td>828.25</td>
<td>0.0118</td>
<td>131.4</td>
<td>140.7</td>
</tr>
<tr>
<td>Yamada Rayleigh Model with Rayleigh curve [41]</td>
<td>459.1</td>
<td>0.0273</td>
<td>28.23</td>
<td>268.4</td>
</tr>
<tr>
<td>Yamada Weibull Model with Weibull curve [42]</td>
<td>565.35</td>
<td>0.0196</td>
<td>57.87</td>
<td>122.1</td>
</tr>
<tr>
<td>Huang Logistic model [16]</td>
<td>394.08</td>
<td>0.0427</td>
<td>10.06</td>
<td>118.6</td>
</tr>
<tr>
<td>Ohba exponential model [3]</td>
<td>4555.4</td>
<td>0.0267</td>
<td>27.09</td>
<td>206.9</td>
</tr>
<tr>
<td>Infection S shaped model [3]</td>
<td>389.1</td>
<td>0.0935</td>
<td>8.69</td>
<td>133.5</td>
</tr>
<tr>
<td>Delayed S shaped model [2]</td>
<td>374.05</td>
<td>0.1976</td>
<td>4.48</td>
<td>168.7</td>
</tr>
<tr>
<td>G-O Model [1]</td>
<td>760</td>
<td>0.0322</td>
<td>112.2</td>
<td>139.8</td>
</tr>
<tr>
<td>Delayed S shaped model with Rayleigh [16]</td>
<td>333.14</td>
<td>0.1004</td>
<td>6.93</td>
<td>798.5</td>
</tr>
</tbody>
</table>

Fig.4. Relative Error Graph SRGM with Generalized Gompertz Testing Effort for Dataset1.

Fig.5. Relative Error Graph SRGM with Generalized Gompertz Testing Effort for Dataset2.
Table 3 describes different testing effort functions and their relative performance metrics for the dataset 2. Table 4 describes various models and their respective parameters and performance metrics like AE and MSE for the dataset 2. The software reliability growth model with proposed testing effort has a=134.23, b=0.153, AE=28.60 and MSE=48.20. From the Table 2 model with proposed testing effort had less AE and MSE values compared with other models.

Figure 5 shows the relative error curve for the dataset 2, which specifies how far our proposed model follows the actual failure phenomenon.

Table 3. Performance Analysis of Proposed Testing Effort Function and other Functions for Dataset2.

<table>
<thead>
<tr>
<th>TEF</th>
<th>Bias</th>
<th>Variation</th>
<th>RMS-PE</th>
<th>PE (at the ending of the testing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponentiated-Gompertz</td>
<td>0.038</td>
<td>0.358</td>
<td>0.351</td>
<td>0.171</td>
</tr>
<tr>
<td>Exponentiated Weibull</td>
<td>0.07</td>
<td>0.387</td>
<td>0.384</td>
<td>0.127</td>
</tr>
<tr>
<td>Burr type X</td>
<td>0.155</td>
<td>0.50</td>
<td>0.512</td>
<td>-0.47</td>
</tr>
<tr>
<td>Yamada exponential</td>
<td>-16.53</td>
<td>6.35</td>
<td>17.71</td>
<td>-13.29</td>
</tr>
<tr>
<td>Yamada Rayleigh</td>
<td>-1.149</td>
<td>3.46</td>
<td>3.64</td>
<td>6.03</td>
</tr>
<tr>
<td>Huang logistic</td>
<td>0.055</td>
<td>0.35</td>
<td>0.35</td>
<td>-0.10</td>
</tr>
</tbody>
</table>

Table 4. Performance Analysis of Proposed Software Reliability Model and other Models for Dataset2.

<table>
<thead>
<tr>
<th>MODEL</th>
<th>a</th>
<th>b</th>
<th>AE(%)</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM with eq. proposed TEF eq . 13</td>
<td>134.23</td>
<td>0.153</td>
<td>28.60</td>
<td>48.20</td>
</tr>
<tr>
<td>SRGM with [14] Exponentiated Weibull</td>
<td>133.87</td>
<td>0.154</td>
<td>28.79</td>
<td>78.55</td>
</tr>
<tr>
<td>SRGM with Burr type X[15]</td>
<td>134.11</td>
<td>0.152</td>
<td>28.67</td>
<td>123.7</td>
</tr>
<tr>
<td>Yamada Rayleigh Model with [41] with Rayleigh curve</td>
<td>866.94</td>
<td>0.00962</td>
<td>25.11</td>
<td>89.24</td>
</tr>
<tr>
<td>Huang Logistic model [16]</td>
<td>138.02</td>
<td>0.1451</td>
<td>26.58</td>
<td>62.41</td>
</tr>
<tr>
<td>Ohba exponential model[3]</td>
<td>137.2</td>
<td>0.156</td>
<td>27.12</td>
<td>3019.6</td>
</tr>
<tr>
<td>Infection S shaped model[3]</td>
<td>159.11</td>
<td>0.0765</td>
<td>15.36</td>
<td>118.3</td>
</tr>
<tr>
<td>Delayed S shaped model[2]</td>
<td>237.2</td>
<td>0.0963</td>
<td>26.16</td>
<td>245.24</td>
</tr>
<tr>
<td>G-O Model[1]</td>
<td>142.32</td>
<td>0.1246</td>
<td>24.29</td>
<td>2438.3</td>
</tr>
</tbody>
</table>

Fig.6. Relative Error Graph SRGM with Generalized Gompertz Testing Effort for Dataset3.

Table 5 describes different testing effort functions and their relative performance metrics for the dataset 3.

Table 5. Performance Analysis of Proposed Testing Effort Function and other Functions for Dataset3.

<table>
<thead>
<tr>
<th>TEF</th>
<th>Bias</th>
<th>Variation</th>
<th>RMS-PE</th>
<th>PE (at the ending of the testing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponentiated-Gompertz</td>
<td>-0.008</td>
<td>2.179</td>
<td>2.129</td>
<td>1.273</td>
</tr>
<tr>
<td>Exponentiated Weibull</td>
<td>0.058</td>
<td>2.212</td>
<td>2.156</td>
<td>1.288</td>
</tr>
<tr>
<td>Burr type X</td>
<td>0.155</td>
<td>2.35</td>
<td>2.35</td>
<td>-0.457</td>
</tr>
<tr>
<td>Yamada Weibull</td>
<td>0.182</td>
<td>2.37</td>
<td>2.37</td>
<td>-0.44</td>
</tr>
<tr>
<td>Yamada Rayleigh</td>
<td>0.324</td>
<td>2.38</td>
<td>2.40</td>
<td>0.15</td>
</tr>
<tr>
<td>Huang logistic</td>
<td>-0.122</td>
<td>2.28</td>
<td>2.28</td>
<td>0.19</td>
</tr>
</tbody>
</table>
Table 6 describes various models and their respective parameters and performance metrics like AE and MSE for the dataset 3. The software reliability growth model with proposed testing effort has $a=94.89$, $b=0.025$ and $MSE=5.42$. From the Table 2 model with proposed testing effort had less AE and MSE values compared with other models.

<table>
<thead>
<tr>
<th>MODEL</th>
<th>$a$</th>
<th>$r$</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM with eq. 13 proposed TEF</td>
<td>94.89</td>
<td>0.025</td>
<td>5.42</td>
</tr>
<tr>
<td>SRGM with Exponentiated Weibull [14]</td>
<td>94.8</td>
<td>0.025</td>
<td>5.70</td>
</tr>
<tr>
<td>SRGM with eq Burr type X [14]</td>
<td>94.61</td>
<td>0.0254</td>
<td>6.55</td>
</tr>
<tr>
<td>Yamada Weibull Model with Weibull curve [42]</td>
<td>87.03</td>
<td>0.0345</td>
<td>7.77</td>
</tr>
<tr>
<td>Huang Logistic model [16]</td>
<td>88.89</td>
<td>0.0390</td>
<td>25.2</td>
</tr>
<tr>
<td>Delayed S shaped model [2]</td>
<td>88.65</td>
<td>0.028</td>
<td>6.31</td>
</tr>
<tr>
<td>G-O Model [1]</td>
<td>137.1</td>
<td>0.0515</td>
<td>25.3</td>
</tr>
</tbody>
</table>

Figure 6 shows the relative error curve for the dataset 3, which specifies how far our proposed model follows the actual failure phenomenon.

6. Optimal Release Time Policy

Identifying the exact time of release of software product from the software industry is an important issue. [7-9, 15] Many authors have focused on this issue. Determining the exact time of release of software product can greatly impact on the software cost and its quality. If the software product released in a short time can save time and cost but we have to compromise with reliability; if the product released after a long and through testing, the product can have quality but it increases the cost of testing. Release time of the software product depends on two factors, one reliability and another cost. So, we need to determine the time of delivery of software product such that the product can have high reliability and cost of expenditure of testing is minimized.

6.1. Software Reliability

Software reliability is defined as probability of product should function correctly in a given environment before it fails.

$$R = e^{-[m(t+\Delta t) - m(t)]}$$

(25)

In the equation (25) $R$ represents the reliability of software product. Figure 7 represents the estimated reliability of software product from the proposed model. In the equation 25 $m(t)$ is mean value of proposed software reliability model. $\Delta t$ represents the fraction of time period, in our case we assumed the value 0.2.

![Fig.7. Estimated Reliability Curve for the Dataset1.](image)

6.2. Software Cost

Computing the software cost during the software development was is an important task, several authors have concentrated much on this issue. Cost of the software product development can be an integration of different cost’s, one before testing phase, one during the testing and cost related to after delivery. Let’s assume $C_1$ is the cost incur for finding the errors pre testing phase, $C_2$ is cost to detect the errors during the testing phase and $C_3$ is cost incur to find errors after the testing phase.
The term \( \text{COST}(t) \) in the equation (26) represents the total cost incurs during product development. \( T \) represents the constant usually we assume maximum time for this quantity, in our paper we assign the value 100 to this term.

\[
\text{COST}(t_i) = C_1 m(t_i) + C_2 [m(T) - m(t_i)] + C_3 \int_{0}^{T} w(t_i) \, dt
\]  

(26)

From the equation (26) we computed the cost of development by assigning \( C_1=1\$ \), \( C_2=5\$ \) and \( C_3=10\$ \) for the dataset-1. Figure 8 shows the graph of computed cost for the proposed software reliability growth model. Table 7 shows the computed cost and reliability of proposed software reliability model.

### 6.3. Release Time Determination

From the equation (25) and (26) we have computed reliability and cost of software product. Software product can be released into market either time at which cost of product become optimal or time at which reliability of software become standard accepted level. In this paper standard acceptance of software reliability level is 0.95 which can be obtained at \( T_R = 38.5 \) weeks and optimal cost \( \text{COST} = 1120\$ \) was obtained at \( T_C = 40 \) weeks.

\[
T_0 = \max(T_R, T_C)
\]  

(27)

From the equation (27) we can extract the optimal time \( T_0 \) which satisfies the our both conditions. In our case the optimal release time \( T_0 = 38.5 \) weeks.

### Table 7. Cost and Reliability for the Dataset 1.

<table>
<thead>
<tr>
<th>TIME</th>
<th>RELIABILITY</th>
<th>COST</th>
<th>TIME</th>
<th>RELIABILITY</th>
<th>COST</th>
<th>TIME</th>
<th>RELIABILITY</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>0.03</td>
<td>1456.36</td>
<td>21</td>
<td>0.18</td>
<td>1179.84</td>
<td>31</td>
<td>0.66</td>
<td>1124.67</td>
</tr>
<tr>
<td>12</td>
<td>0.03</td>
<td>1413.67</td>
<td>22</td>
<td>0.21</td>
<td>1167.97</td>
<td>32</td>
<td>0.72</td>
<td>1123.53</td>
</tr>
<tr>
<td>13</td>
<td>0.04</td>
<td>1374.48</td>
<td>23</td>
<td>0.25</td>
<td>1158.11</td>
<td>33</td>
<td>0.77</td>
<td>1122.67</td>
</tr>
<tr>
<td>14</td>
<td>0.05</td>
<td>1338.8</td>
<td>24</td>
<td>0.29</td>
<td>1150.02</td>
<td>34</td>
<td>0.81</td>
<td>1122.03</td>
</tr>
<tr>
<td>15</td>
<td>0.06</td>
<td>1306.61</td>
<td>25</td>
<td>0.33</td>
<td>1143.45</td>
<td>35</td>
<td>0.85</td>
<td>1121.56</td>
</tr>
<tr>
<td>16</td>
<td>0.07</td>
<td>1277.83</td>
<td>26</td>
<td>0.38</td>
<td>1138.18</td>
<td>36</td>
<td>0.89</td>
<td>1121.22</td>
</tr>
<tr>
<td>17</td>
<td>0.08</td>
<td>1252.35</td>
<td>27</td>
<td>0.44</td>
<td>1134.00</td>
<td>37</td>
<td>0.92</td>
<td>1120.97</td>
</tr>
<tr>
<td>18</td>
<td>0.1</td>
<td>1230.01</td>
<td>28</td>
<td>0.49</td>
<td>1130.71</td>
<td>38</td>
<td>0.94</td>
<td>1120.58</td>
</tr>
<tr>
<td>19</td>
<td>0.12</td>
<td>1210.62</td>
<td>29</td>
<td>0.55</td>
<td>1128.15</td>
<td>39</td>
<td>0.96</td>
<td>1120.68</td>
</tr>
<tr>
<td>20</td>
<td>0.15</td>
<td>1193.97</td>
<td>30</td>
<td>0.61</td>
<td>1126.18</td>
<td>40</td>
<td>0.97</td>
<td>1120.6</td>
</tr>
</tbody>
</table>

### 7. Conclusion

In this paper we have investigated the testing effort dependent software reliability growth model by incorporating new proposed Exponentiated-Gompertz testing effort. Exponentiated-Gompertz testing effort function is associated with various characteristics of failure rates and can be suitable for fitting any sort of dataset. Software reliability growth model with proposed testing effort function gives excellent fit and captures the real testing effort perfectly. Performance metrics were used to compare the proposed model with previous models, results shows that proposed testing effort model have satisfactory results. In this paper we have computer the release time of the software product depending on the quantities like cost and reliability.
Software Reliability Growth Models with Exponentiated-gompertz Testing Effort and Release Time Determination

IJECECS, vol.6, no.2, pp.48-55, 2014.DOI: 10.5815/ijmecs.2014.02.07


Authors’ Profiles


Dr. Shaheda Akhtar received Bachelor of Computer Science and Master of Computer Science from Acharya Nagarjuna University, M.S from B.I.T.S Pilani. Ph.D from Acharya Nagarjuna University. Presently working as Head of the department of Computer Science, Government College for Women (A), Guntur, A.P, India and Research Director for Department of Computer Science & Engineering, Acharya Nagarjuna University, Guntur, A.P, India. Areas of interest are Software Engineering, reliability and quality control, Software Architecture Recovery, Machine Learning and Data Mining. Published more than 35 research papers in various international journals.