Software Reliability Growth Model With Testing-Effort by Failure Free Software

K.Venkata Subba Reddy, Dr.B.Raveendrababu

Abstract—Software reliability is the probability that a given software system in a given environment will operate correctly for a specified period of time. Software reliability engineering is focused on engineering techniques for developing and maintaining software systems whose reliability can be quantitatively evaluated. In order to estimate as well as predict the reliability of software systems, failure data need to be properly measured by various means during software development and operational phases. Moreover, credible software reliability models are required to track underlying software failure processes for accurate reliability analysis and forecasting. In particular, vital future goals include the development of new software reliability engineering paradigms that take software architectures, testing techniques, and software failure manifestation mechanisms into consideration. In the present paper we study the performance of the Bayes Shrinkage estimators for the scale parameter of the Weibull distribution. We will consider the case where the time dependent behaviors of testing effort expenditures are described by New Modified Weibull Distribution (NMWD). Software Reliability Growth Models (SRGM) based on the NHPP are developed which incorporates the (NMWD) testing-effort expenditure during the software testing phase. It is assumed that the error detection rate to the amount of testing-effort spent during the testing phase is proportional to the current error content. Model parameters are estimated by Least Square and Maximum Likelihood estimation techniques, and software measures are investigated through numerical experiments on real data from various software projects[32].

Index Terms—Change-Point, Non-homogeneous Poisson Process (NHPP), Software Reliability Growth Model (SRGM), NMWD, Software Testing.

I. INTRODUCTION

Software Reliability Growth Model (SRGM) is a mathematical model of how the software reliability improves as faults are detected and repaired. SRGM can be used to predict when a particular level of reliability is likely to be attained. Thus, SRGM is used to determine when to stop testing to attain a given reliability level. Software reliability is the probability that the given software functions correctly under a given environment during the specified period of time [1], [2], [3], [4]. Therefore, modeling of software reliability accurately and predicting its possible trends are essential for determining overall reliability of the software. Various SRGM have been developed during the last three decades and they can provide very useful information about how to improve reliability [5], [6], [7], [8]. One can be easily determined some important metrics like time period, number of remaining faults, mean time between failures (MTBF), and mean time to failure (MTTF) through SRGM. The ‘time of failure’ and ‘average life’ of a component, measured from some specified time until it fails, is represented by a continuous random variable. Extensively in recent years, one distribution that has been used as a model to deal with such problems for product life is the Weibull distribution. Its applications in life-testing problems and survival analysis have been widely advocated (Weibull, 1951; Berrettoni, 1964). It has been used as model with diverse types of items such as ball bearing (Lieblein & Zelen, 1956), vacuum tube (Kao, 1959) and electrical isolation (Nelson, 1972). Mittnik & Reachev (1993) found that the Weibull distribution might be an adequate statistical model for stock returns. Mann (1968) gave a variety of situations in which the distribution is used for other types of failure data.

II. NEW MODIFIED WEIBULL TEST-EFFORT FUNCTIONS AND CHANGE-POINTS

Whittemore & Altschuler (1976) used it as a model in biomedical applications. The probability density function of the Weibull distribution is given by

$$f(x; \nu, \theta) = \frac{\nu}{\theta} x^{\nu-1} e^{-\left(\frac{x}{\theta}\right)^\nu}, \quad x>0, \nu>0, \theta>0$$

(1)

Where the parameters $\nu$ and $\theta$ are referred to as the shape and scale parameters of the distribution, respectively. The consumed testing-effort indicates how the errors are detected effectively in the software and can be modified by different distributions. Many authors reported that Yamada Weibull-Type testing-effort curves may have an apparent peak phenomenon during the software development process when shape parameter $m = 3, 4,$ or $5$ [17], [18], [19]. Basically, the software reliability is highly related to the amount of testing-effort expenditures spent on detecting and correcting software errors. Therefore, we propose the NMW curve as a more flexible testing-effort function. The cumulative testing effort expenditure consumed in time $(0, t]$ is

$$W(t) = \alpha(1-e^{-\beta t^m \delta^{\nu^m}}), \alpha > 0, \beta > 0, m \geq 0, \delta > 0.$$  

(2)

And the current testing-effort consumed at testing time $t$ is

$$w(t) = \alpha \beta (m + \delta t)^{m-1} e^{\delta t} e^{-\beta (m + \delta t)^m \delta^{\nu^m}}$$

(3)

Where $\alpha, \beta, m, \delta$ are constant parameters, is the total amount of testing-effort expenditures; and are the scale parameters, and $m$ is shape parameter.
III. INCORPORATING TEST-EFFORT FUNCTION INTO SOFTWARE RELIABILITY MODELING

In this section, we will briefly review some testing -effort functions. During software testing phase, it consumes much test-effort, such as man power, number of test cases, and CPU time. Traditionally, the test-effort during the testing phase and the time-dependent behavior of development effort in the software development process can be described by an Exponential, a Rayleigh or a Weibull curve, which were proposed by Yamada et al. [2, 13,14], Musa et al. [15], Putnam [25] and Kapur et al. Let W(t) be the cumulative amount of testing-effort expenditures in the testing time interval (0, t] and g(t) be the consumption rate of the testing effort expenditures. Thus, the testing-effort consumed per unit time is assumed to the proportional to the remaining amount of the testing-effort expenditures a - W(t). Following the general assumptions [26, 27], we can get the differential equation.

\[ \frac{dW(t)}{dt} = g(t) \times [\alpha - W(t)], \alpha > 0 \]  

(4)

Solving the above equation, we get

\[ W(t) = \alpha \times [1 - e^{-\int_{0}^{t} g(x)dx}] \]  

(5)

and W(t) is defined as follows:

\[ W(t) = \int_{0}^{t} w(x)dx \]  

(6)

Where \( \alpha \) is the total amount of testing effort to be eventually consumed, g(t) is the consumption rate of testing-effort expenditures and w(t) is the current testing effort consumed at time t [3].

If g(t) = p, then w(t)= \( \alpha \beta \) exp[-\( \beta t \)], we have an Exponential curve and the integral forms of w(t) (i.e. the cumulative testing-effort consumed in time (0, t]) is

\[ W(t)=\alpha(1-\exp[-\beta t]) \]  

(7)

If g(t) = \( \beta t \), then w(t)= \( \alpha \beta t \) exp[-\( \beta T/t \)] and we have a Raleigh curve and the integral forms of w(t) is

\[ W(t)=\alpha(1-\exp[-\beta T/t]) \]  

(8)

and if w(t)= \( \alpha \beta m \) tm-1exp[-\( \beta t/m \)] we have a Weibull Curve and the integral forms of w(t) is

\[ W(t)=\alpha(1-\exp[-\beta t/m]) \]  

(9)

Where \( \beta \)the scale parameter and m is the shape parameter.

In the Weibull-type curves (i.e. Eq. (9)), when m=1 or m=2, we obtain the exponential or the Rayleigh curve respectively and they are the special cases of the Weibull testing-effort function. In the Weibull-type curves, when m=3, 4, and 5, we can find that these testing-effort curves almost have an apparent peak Phenomenon (i.e. non-smoothly increasing and degrading consumption curve) during the software development process. That is, an extreme peak work rate will occur. This phenomenon seems not so realistic because it is not commonly used to interpret the actual software development test process. It sometimes may not be suitable for modeling the test effort consumption curve although the Weibull function can be made to fit or approximate many distributions and represents flexible testing effort by controlling the shape parameter m. General speaking, from our study [1-4, 7, 22] the estimated value of m usually will not be larger than 2.5 in many real world applications.

IV. REVIEWS OF FAULT DETECTION AND CORRECTION IN SRGM

The fault removal process follows the Nonhomogeneous Poisson Process (NHPP). The software system is subject to failures at random times caused by the manifestation of remaining faults in the system. Each time a failure occurs, the fault that caused it is immediately and perfectly removed. A detected error is removed with certainty and correction of errors takes only negligible time. No new faults are introduced. The fault-detection process follows the NHPP and the rate of change of the mean value function (MVF) is exponentially decreasing. In fact, most existing SRGMs can be reinterpreted as delayed fault-detection models that can model the software fault detection and correction processes. Therefore, we can remove the impractical assumption that the fault-correction process is perfect and establish a corresponding time dependent delay function to fit the fault-correction process. Given a fault-detection and fault-correction process, one defines the delay-effect factor, \( \alpha(t) \), to be a time-dependent function that measures the expected delay in correcting a detected fault at any time. An SRGM is called Delayed-time NHPP model, in this model the fault detection process follows the NHPP. The software system is subject to failures at random times caused by the manifestation of remaining faults in the system. The rate of change of the MVF is exponentially decreasing. The detected faults are not immediately removed and it lags the fault detection process by a delay-effect factor \( \alpha(t) \).

In the paper, we will review three conventional SRGMs that can be directly derived from delayed time NHPP model and delay-effect factor [32].

Goel-Oukomo Model: This model, first proposed by Goel and Okumoto [2, 4], is one of the most popular NHPP model in the field of software reliability modeling

Yamada Delayed S-Shaped Model: The Yamada Delayed S-Shaped model is a modification of the NHPP to obtain an S-shaped curve for the cumulative number of failures detected such that the failure rate initially increases and later decays

Yamada Weibull-Type Testing-Effort Function Model: Yamada et al. [2, 7] proposed a software reliability model incorporating the amount of test-effort expended during the software testing phase. The testing-effort can be represented as the man power, number of CPU hours, or the number of executed test cases, etc. In general, the testing-effort during
The testing phase and the time dependent behavior of development effort in the software development process can be described by a Weibull curve.

V. FAILURE DEPENDENCY IN SOFTWARE FAULT MODELLING

Let \( a \) is the expected number of initial faults, \( m(t) \) be the MVF of the expected number of faults detected in time \( (0, t] \), and \( r \) is the fault detection rate. Where \( \theta \) is the dependent fault removal rate. Dependent faults can be removed only when the leading fault is perfectly removed. The Mean Value Function MVF is given by

\[
m(t) = a (1 - P(1 + r)) \exp[-\theta t] - (1 - P) \times \exp\left[\frac{2 \theta}{\nu} (1 - \exp[-\nu t]) - \nu \theta (1 + \exp[-\nu t])\right].
\]

(10)

VI. NUMERICAL EXAMPLE

A. Preliminaries

We choose two real data sets as illustrations. The first data set (DS1) was from a study by Ohba [9]. The system was a PL/I database application software consisting of approximately 1,317,000 lines of code. During nineteen weeks, 47.65 CPU hours were consumed and about 328 software faults were removed. The second data set (DS2), during the test phase, 230 software faults was removed. The complete failure data is given in Table 1.

Table 1: Real Software Failure Data Set

<table>
<thead>
<tr>
<th>Week</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNF</td>
<td>44</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Week</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>CNF</td>
<td>124</td>
<td>130</td>
<td>130</td>
<td>159</td>
<td>175</td>
<td>181</td>
<td>197</td>
<td>205</td>
</tr>
<tr>
<td>Week</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
</tbody>
</table>

DataSet-1: Firstly, all parameters of the proposed models are estimated by using the method of least squares estimation (LSE) or maximum likelihood estimation (MLE) [3-4, 6, 27]. Table 2 shows the estimated parameters of Eq. (10) and the performance comparisons of different SRGMs for DataSet-1. It is noted that the proposed model (i.e., Eq. (10)) estimates \( P=0.81 \) for this data set. The result suggests that the software may contain two categories of faults, 81% are leading faults and 19% are dependent faults. Moreover, the possible values of \( \theta \) are also discussed and listed in Table 2. As seen from Table 2, the proposed model almost provides the lowest MEOP if compared to the Goel-Okumoto model and the Yamada delay S-shaped model. Overall, the MVF of proposed model provides a good fit to this data.

B. Criteria for model’s comparison

The comparison criteria we use to compare various models’ performance are described as follows:

The Noise is defined as [29]:

\[
\sum_{i=1}^{n} \left| \frac{r_i - r_{i-1}}{r_{i-1}} \right|,
\]

(11)

where \( r_i \) is the predicted failure rate.

The Mean Square of Fitting Error (MSE) is defined as:

\[
\sum_{i=1}^{n} (m(t_i) - m_{i-1})^2 / k,
\]

(12)

where \( m_i \) is the observed number of faults by time \( t_i \).

The Mean Error of Prediction (MEOP) is defined as [27]:

\[
\sum_{i=k}^{n} \left| n_i - m_i / (n-k+1) \right|,
\]

(13)

Where \( n_i \) is the observed cumulative number of failures at time \( s_i \) and \( m_i \) is the predicted cumulative number of failures at time \( s_i \), i=k, k+1, ..., n.

TABLE 2: Performance Comparison of different Software Reliability growth Models for Dataset -1

<table>
<thead>
<tr>
<th>Model</th>
<th>( a )</th>
<th>( r )</th>
<th>( \theta )</th>
<th>( \nu )</th>
<th>Mean Square of Fitting Error (MSE)</th>
<th>The Mean Error of Prediction (MEOP)</th>
<th>Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Model Eq 10</td>
<td>412.600</td>
<td>0.228869</td>
<td>0.090558</td>
<td>0.81</td>
<td>161.585</td>
<td>9.75609</td>
<td>2.16918</td>
</tr>
<tr>
<td>Proposed Model Eq 10</td>
<td>523.048</td>
<td>0.460176</td>
<td>0.283841</td>
<td>0.2</td>
<td>139.241</td>
<td>9.68111</td>
<td>1.50403</td>
</tr>
<tr>
<td>Proposed Model Eq 10</td>
<td>501.357</td>
<td>0.358750</td>
<td>0.192887</td>
<td>0.3</td>
<td>145.669</td>
<td>9.79057</td>
<td>1.69780</td>
</tr>
</tbody>
</table>
**DataSet-2:** Similarly, parameters of all selected models are estimated and the related MVFs are obtained. All selected models are compared with each other based on objective criteria. Table 3 shows the estimated parameters of Eq. (10) and the performance comparisons of different SRGMs for Dataset 2. The proposed model estimates \( P=0.83 \) and it indicates that the software contains two categories of faults, 83% are leading faults and 17% are dependent faults. Moreover, the possible values of \( P \) is also listed in Table 3. On the other hand, we know that the inflection S-shaped model is based on the dependency of faults by postulating the assumption: some of the faults are not detectable before some other faults are removed [31]. Therefore, it may provide us some information for reference. After the simulation, we find that the estimated value of inflection rate (which indicates the ratio of the number of detectable faults to the total number of faults in the software) is 0.834 for Dataset 2. It indicates that the growth curve is slightly S-shaped [12-13]. On the average, the proposed model performs well in this actual data.

<table>
<thead>
<tr>
<th>Model</th>
<th>( a )</th>
<th>( r )</th>
<th>( \theta )</th>
<th>( P )</th>
<th>Mean Square of Fitting Error (MSE)</th>
<th>The Mean Error of Prediction (MEOP)</th>
<th>Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Model Eq 10</td>
<td>264.181</td>
<td>0.218560</td>
<td>0.082480</td>
<td>0.83</td>
<td>402.515</td>
<td>13.2156</td>
<td>2.83969</td>
</tr>
<tr>
<td>Proposed Model Eq 10</td>
<td>330.303</td>
<td>0.683468</td>
<td>0.246725</td>
<td>0.2</td>
<td>334.808</td>
<td>14.6244</td>
<td>1.68513</td>
</tr>
<tr>
<td>Proposed Model Eq 10</td>
<td>313.562</td>
<td>0.409470</td>
<td>0.184855</td>
<td>0.3</td>
<td>370.289</td>
<td>14.2028</td>
<td>2.00071</td>
</tr>
<tr>
<td>Proposed Model Eq 10</td>
<td>301.219</td>
<td>0.325860</td>
<td>0.144944</td>
<td>0.4</td>
<td>383.379</td>
<td>13.9400</td>
<td>2.24942</td>
</tr>
<tr>
<td>Proposed Model Eq 10</td>
<td>290.181</td>
<td>0.281493</td>
<td>0.1191115</td>
<td>0.5</td>
<td>390.808</td>
<td>13.7258</td>
<td>2.42998</td>
</tr>
<tr>
<td>Proposed Model Eq 10</td>
<td>279.888</td>
<td>0.251092</td>
<td>0.101618</td>
<td>0.6</td>
<td>396.008</td>
<td>13.5182</td>
<td>2.59620</td>
</tr>
<tr>
<td>Proposed Model Eq 10</td>
<td>270.325</td>
<td>0.230121</td>
<td>0.089176</td>
<td>0.7</td>
<td>400.082</td>
<td>13.3320</td>
<td>2.74551</td>
</tr>
<tr>
<td>Proposed Model Eq 10</td>
<td>261.688</td>
<td>0.214181</td>
<td>0.079991</td>
<td>0.8</td>
<td>403.478</td>
<td>13.1691</td>
<td>2.99694</td>
</tr>
<tr>
<td>Proposed Model Eq 10</td>
<td>253.983</td>
<td>0.201489</td>
<td>0.072878</td>
<td>0.9</td>
<td>406.418</td>
<td>13.0372</td>
<td>2.99694</td>
</tr>
<tr>
<td>Goel-Okumoto model</td>
<td>326.364</td>
<td>0.055693</td>
<td>-</td>
<td>-</td>
<td>253.217</td>
<td>12.7256</td>
<td>1.35427</td>
</tr>
<tr>
<td>Yamada Delay S-shaped model</td>
<td>247.221</td>
<td>0.091014</td>
<td>-</td>
<td>-</td>
<td>409.026</td>
<td>12.9325</td>
<td>3.10483</td>
</tr>
</tbody>
</table>
VII. CONCLUSION

In this paper, we have proposed a SRGM incorporating the testing-effort function that is completely different from the traditional Weibull-type curve. We will review three conventional SRGMs that can be directly derived from delayed time NHPP model and delay-efffect factor. Some new SRGMs are proposed and several numerical illustrations based on two real data sets are presented. Experimental results show that the proposed framework to incorporate both failure dependency and time-dependent delay function for SRGM has a fairly accurate prediction capability. However, the above statements imply that our model still have better predictive validity based on real failures experienced and give a more accurate prediction.

REFERENCES


AUTHORS PROFILE

K.VenKata Subba Reddy obtained his B.Tech in Information Technology from University of Madras in 2002 and received the M.Tech in Software Engineering from Bharath University, Chennai in 2005. He is pursuing Ph.D. in Computer Science and Engineering, under the guidance of Dr.B.Raveendra Babu, at Acharya Nagarjuna University, Guntur, Andhra Pradesh, India. He is currently working as Assistant Professor in Computer Science & Engineering Department at Muffakham Jah College of Engineering & Technology, Banjara hills, and Hyderabad. He has 9 years of experience in teaching. His research interests include Software Engineering, Software Quality Assurance and Testing and Cloud computing. He is a life member of ISTE and a member of CSI.

Dr B. Raveendra Babu obtained his Masters in Computer Science and Engineering from Anna University, Chennai. He received his Ph.D. at S.V University, Tirupati. He is currently leading a Team as Director Vignana Jyothi Institute of Management, Hyderabad. He has 26 years of teaching experience. He has more than 25 international & national publications to his credit. His research areas of interest include VLDB, Image Processing, Pattern analysis and Information.