Unified Framework For Developing Testing Effort Dependent Software Reliability Growth Models With Change Point And Imperfect Debugging

V.S.S. Yadavalli¹, Anu G. Aggarwal², P.K. Kapur³ and Jyotish Kumar⁴

¹Department of Industrial and Systems Engineering, University of Pretoria, Pretoria, South Africa
²,³,⁴Department of Operational Research, University of Delhi, Delhi-110007

Email: ¹sarman.yadavalli@up.ac.za; ²anuagg17@gmail.com; ³pkkapurl@gmail.com

ABSTRACT
In order to address the continuing demand for high quality, low cost, reliable software, hundreds of software reliability growth models (SRGMs) have been proposed in recent years. In spite of the diversity and elegance of many of these, no single model can be readily recommended as best to represent the challenging nature of the software testing. Of late, some authors have tried to develop a unifying approach so as to capture different growth curves, thus easing the model selection process. The work in this area done so far relates the fault removal process to the testing / execution time and does not consider the consumption pattern of resources such as computer time, manpower and number of executed test cases etc. More realistic modeling can result if the reliability growth process is studied with respect the amount of expended testing efforts. Due to the complexity of software system and incomplete understanding of software, the testing process may not be perfect or the fault detection /correction rate may change at any time moment. In this paper, we propose a generalized framework for deriving several existing as well as new testing effort dependent software reliability growth models incorporating change point and the possibility of imperfect debugging and error generation. The proposed framework is based on standard probability distribution functions. The models developed have been validated and verified using real data sets. Estimated Parameters and comparison criteria results have also been presented.

KEYWORDS
Non-homogenous Poisson process, Software Reliability Growth Model, Hazard Rate, Imperfect Debugging, Change-point.

INTRODUCTION
The concept of "software reliability" and its measurement is receiving a lot of attention in the software development community. With the ever increasing role that software is playing in today's and tomorrow's world, the software developers and users are asking: "Just how 'good' is the software?" and "How much testing should be done before the software is released?" The software reliability methodology attempts to provide quantitative measures to help answer these questions. Software reliability is one of the important parameters of software quality and system dependability. It is defined as the probability of failure-free software operation in a specified environment for a specified period of time.

The development of high quality software satisfying cost, schedule and resource requirement is an essential prerequisite for improved competitiveness of any organization. One major difficulty to master this challenge is the inevitability of defects in software products. The testing of software systems is subject to strong conflicting forces. One of the most effective ways to do this is to apply software reliability engineering to testing (and development). Software reliability engineering delivers the desired functionality for a product much more efficiently by quantitatively characterizing its expected use. The software reliability engineering tends to increase reliability while decreasing development time and cost. Thus software reliability engineering balances customer needs for the major quality characteristics of reliability, availability, delivery time and life cycle cost more effectively.

Hundreds of Software Reliability Growth models (SRGMs), which relate the number of failures (faults identified/corrected) and execution time, have been discussed in the software reliability engineering literature. Most of the existing growth models belong to either of following two categories: Exponential Goel and Okumoto [1] and S-shaped Yamada and Osaki [18], Obha [8]. According to their category, they provide fit on the different types of failure Data sets. Later attempts were made to present the flexible SRGMs which can work on both types of failure datasets e.g. Obha [7], Bittanti et al. [17] and Kapur and Garg models [13].

In most of the models discussed above it is assumed that whenever an attempt is made to remove a fault, it is removed with certainty i.e. a case of perfect debugging. But the debugging activity is not always perfect because of number of factors like tester’s skill/expertise, complexity of the software etc. The testing team may not be able to remove/correct fault perfectly on observation/detection of a failure and the original fault may remain leading to a phenomenon known as imperfect debugging, or replaced by another fault resulting in error generation. In case of imperfect debugging the fault content of the software is not changed, but because of incomplete understanding of the software, the original detected fault is not removed perfectly. However, in case of error generation, the total fault content increases as the testing progresses because...
new faults are introduced in the system while removing the old original faults. Model due to Obha and Chou [6] is an error generation model applied on G-O model and has been also named as Imperfect debugging model. Kapur and Garg [14,15] introduced the imperfect debugging in G-O model. They assumed that the FDR per remaining faults is reduced due to imperfect debugging. Thus the number of failures observed/detected by time infinity is more than the initial fault content. We execute the program in specific environment and improve its quality by detecting and correcting faults. Many SRGM assume that, during the fault detection process, each failure caused by a fault occurs independently and randomly in time according to the same distribution Musa et al. [4]. But the failure distribution can be affected by many factors such as running environment, testing strategy, defect density and resource allocation. In practice, if we want to detect more faults for a short period of time, we may introduce new techniques or tools that are not yet used, or bring in consultants to make a radical software risk analysis. Thus, the fault detection rate may not be smooth and can be changed at some time moment τ called change-point. Many researchers have incorporated change point in software reliability growth modeling. Firstly Zhao [9] incorporated change-point in software and hardware reliability. Huang et al. [20] used change-point in software reliability growth modeling with testing effort functions. The imperfect debugging with change-point has been introduced in software reliability growth modeling by Shyur [3]. Kapur et al. [11,12] introduced various testing effort functions and testing effort control with change-point in software reliability growth modeling. These SRGM have been developed for diverse testing environment like testing effort expenditure, imperfect debugging, change-point etc. But no SRGM can be claimed to be the best as the physical interpretation of the testing and debugging changes due to numerous factors e.g., design of test cases, defect density, skills and efficiency of testing team, availability of testing resources etc. The plethora of SRGM makes the model selection a tedious task. To reduce this difficulty, unified modeling approaches have been proposed by many researchers. These schemes have proved to be successful in obtaining several existing SRGM by following single methodology and thus provide an insightful investigation for the study of general models without making many assumptions [10].

In this paper, we present a unified framework for Software reliability growth modeling with respect to testing effort expenditure and incorporate the concept of change point with imperfect debugging and error generation. This unified scheme is based on Probability distribution functions. It is also shown that previously reported Non-Homogeneous Poisson Process (NHPP) based SRGMs with imperfect debugging and error generation are special cases of the proposed framework. From this approach, we can not only obtain existing models but also develop some new NHPP models.

The existing and new models derived here have been validated and evaluated on two actual software failure data sets. Non-linear regression based on least square method has been used for Parameter estimation and MSE (Mean Squared Error) and R² has been used as the comparison criteria. The goodness of fit curves have been drawn to illustrate the fitting of the models to the data graphically.

NOTATIONS

\( m(W_t) \): The mean value function or the expected number of faults detected or removed by time \( t \).

\( a(W_t) \): Total fault content of software dependent on time.

\( p \): The probability of fault removal on a failure (i.e., the probability of perfect debugging).

\( a \): The rate at which the faults/errors may be introduced during the debugging process.

\( b \): Fault removal/correction rate.

\( \lambda(W_t) \): Intensity function for NHPP models or fault detection rate per unit time.

\( F(W_t) \): Distribution functions for fault removal/correction times.

\( f(W_t) \): Density functions for fault removal/correction times.

\( s(W_t) \): Hazard rate function.

\( \beta \): Learning parameter in logistic function.

\( \tau \): Change Point.

PROPOSED UNIFIED MODELING FRAMEWORK

BASIC ASSUMPTION

The NHPP models are based on the assumption that the software system is subject to failures at random times caused by manifestation of remaining faults in the system. Hence NHPP are used to describe the failure phenomenon during the testing phase. The counting process \( \{N(t), t \geq 0\} \) of an NHPP process is given as follows.

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

and

\[
m(t) = \int_{a}^{t} \lambda(x) \, dx
\]

The intensity function \( \lambda(x) \) (or the mean value function \( m(t) \)) is the basic building block of all the NHPP models existing in the software reliability engineering literature.
Unified Framework For Developing Testing Effort Dependent Software Reliability Growth Models With Change Point and Imperfect Debugging

The proposed models are based upon the following basic assumptions:

1. Failure fault removal phenomenon is modeled by NHPP.
2. Software is subject to failures during execution caused by faults remaining in the software.
3. Failure rate is equally affected by all the faults remaining in the software.
4. When a software failure occurs, an instantaneous repair effort starts and the following may occur:
   a) Fault content is reduced by one with probability $p$
   b) Fault content remains unchanged with probability $1-p$.
5. During the fault removal process, whether the fault is removed successfully or not, new faults are generated with a constant probability $\alpha$.
6. Fault detection/removal rate may change at any time moment $\tau$.

Assumption 4 and 5 captures the effect of imperfect debugging and error generation respectively.

MODEL DEVELOPMENT

Let the counting processes $\{X(W_t), t \geq 0\}$ and $\{N(W_t), t \geq 0\}$ represent the cumulative number of failures observed and faults corrected up to time $t$ respectively and let the test begun at time $t=0$. Then the distribution of $N(W_t)$ is given by

$$\Pr\{N(W_t) = n\} = \sum_{j=0}^{\infty} \Pr\{X(0) = j\} \Pr\{X(0) = j\}$$

(3)

Here it can be noted that the conditional probability $\Pr\{N(W_t) = n\} | X(0) = j\} = \Pr\{X(0) = j\}$ is zero for $j < n$. For $j \geq n$ it is given by

$$\Pr\{N(W_t) = n\} | X(0) = j\} = \left(\frac{j}{n}\right) (F(W_t))^n (1 - F(W_t))^{j-n}$$

(4)

Therefore, we have

$$\Pr\{N(W_t) = n\} = \sum_{j=0}^{\infty} \left(\frac{j}{n}\right) (F(W_t))^n (1 - F(W_t))^{j-n} a^j \exp(-a) \frac{a^j}{j!}$$

Or we can write

$$\Pr\{N(W_t) = n\} = \left(\frac{a F(W_t)}{n}\right)^n \exp(-a) \sum_{j=0}^{\infty} \left[ a(1 - F(W_t)) \right]^{j-n} \frac{1}{(j-n)!}$$

Or we can write

$$\Pr\{N(W_t) = n\} = \left(\frac{a F(W_t)}{n}\right)^n \exp\left(-a\left[ F(W_t) \right] \right)$$

(5)

Hence we can conclude that the fault correction process is poisson with mean value function (MVF) as given by:

$$m(W_t) = E[N(W_t)] = a F(W_t)$$

(6)

As specified before, here $F(W_t)$ is the testing effort dependent probability distribution function for fault correction times. It can be noted that $F(W_t)$ so defined satisfy all the properties of probability distribution functions.

1. At $t=0, W_t = 0$ and $F(W_t) = 0$. In this paper, we have used three types of testing effort function namely Exponential, Rayleigh and Weibull type. All these functions satisfy the property that at $t=0, W_t = 0$. It can be verified from their expressions, discussed in detail in appendix at the end of the paper.
2. For $t > 0, W_t > 0$ and $F(W_t) > 0$.
3. In this paper we have assumed $F(W_t)$ to be either of Exponential, Erlang, Logistic type. As $t$ increases, $W_t$ also increases indicating monotonically increasing nature of $F(W_t)$ Similarly the continuity of $F(W_t)$ can also be explained.
4. As testing continues for an infinitely large time i.e. $t \rightarrow \infty, W_t \rightarrow \bar{W}$, the corresponding value of distribution function $F(W_t)$ is $F(\bar{W})$. Here $\bar{W}$ is a very large positive number representing the upper bound on the availability of testing resources. Therefore, $F(\bar{W})$ can be assumed to be of order 1.

From Equation (6), the instantaneous failure intensity function $\lambda(W_t)$ is given by:

$$\lambda(W_t) = aF'(W_t)$$

Or we can write
Let us define \( s(W_t) = \frac{F'(W_t)}{1 - F(W_t)} \)

Here \( s(W_t) \) represents hazard rate function or fault detection/correction rate per remaining fault of the software, or the rate at which the individual faults manifest themselves as failures during testing.

Now, Equation (7) can be written as:

\[
\frac{dm}{dt} = \frac{d}{dt}\frac{d}{dt}\left[ a - m(W_t) \right] \frac{F'(W_t)}{1 - F(W_t)} = \left[ a - m(W_t) \right] s(W_t)
\]

**MODEL DEVELOPMENT WITH CHANGE POINT**

Incorporating change-point concept, \( s(W_t) \) i.e. fault detection/correction rate per remaining fault of the software can be written as

\[
s(W_t) = \begin{cases} F'_1(W_t) \\ 1 - F_1(W_t) \\ \frac{F'_2(W_t)}{1 - F_2(W_t)} \end{cases} \quad t \leq \tau
\]

\[
s(W_t) = \begin{cases} \frac{F'_1(W_t)}{1 - F_1(W_t)} \\ F'_2(W_t) \\ \frac{1 - F_2(W_t)}{1 - F_2(W_t)} \end{cases} \quad t > \tau
\]

where \( F'_1(W_t) = \frac{d}{dt}F_1(W_t) \) and \( F'_2(W_t) = \frac{d}{dt}F_2(W_t) \)

Further, incorporating the change-point concept in modeling, Equation (8) becomes:

For \( 0 \leq t \leq \tau \)

\[
\frac{dm}{dt} = \left[ a - m(W_t) \right] \frac{F'_1(W_t)}{1 - F_1(W_t)}
\]

Solving the above equation with initial condition at \( t=0, W_0 = 0 \) and \( m(W_t) = 0 \), we get

\[
m(W_t) = a \cdot F_1(W_t)
\]

For \( t > \tau \)

\[
\frac{dm}{dt} = \left[ a - m(W_t) \right] \frac{F'_2(W_t)}{1 - F_2(W_t)}
\]

Solving above equation with initial condition at \( t=\tau, W_\tau = W_\tau \) and \( m(W_t) = m(W_\tau) \), we get

\[
m(W_t) = a \left[ 1 - \frac{(1 - F_1(W_\tau))(1 - F_2(W_\tau))}{(1 - F_2(W_\tau))} \right]
\]

**MODEL DEVELOPMENT WITH CHANGE POINT & TWO TYPE OF IMPERFECT DEBUGGING**

In this section, we formulate distribution based software reliability growth models incorporating change-point and two types of imperfect debugging. Since the faults in the software systems are detected and eliminated during the testing phase, the number of faults remaining in the software system gradually decreases as the testing procedure go on. Thus under the common assumptions for software reliability growth modeling, we consider the following linear differential equation.

\[
\frac{dm}{dt} = b(W_t)(a - m(W_t))
\]

Where \( b(W_t) \) is a fault detection rate per remaining faults at testing time. Here we consider the fault detection rate as hazard rate \( s(W_t) \), initial fault is not the constant but the function of \( t \) and incorporating the imperfect debugging. So the above equation can be written as
Unified Framework For Developing Testing Effort Dependent Software Reliability Growth Models With Change Point and Imperfect Debugging

\[ \frac{dm}{dt} = s(W_t)p(W_t)(a(W_t) - m(W_t)) \]

(13)

We assume that faults can be introduced during the debugging phase with a constant fault introduction rate \( \alpha \). Therefore, the fault content rate function, \( a(W_t) \), is a linear function of the expected number of faults detected by \( W_t \) and it is defined by:

\[
a(W_t) = \begin{cases} 
  a + \alpha m(W_t) & \text{for } t \leq \tau \\
  a + \alpha m(W_t) + \alpha_2 (m(W_t) - m(W_{\tau})) & \text{for } t > \tau
\end{cases}
\]

and Probability of perfect debugging rate will be

\[
p(W_t) = \begin{cases} 
  p_1 & \text{for } t \leq \tau \\
  p_2 & \text{for } t > \tau
\end{cases}
\]

(14)

Now using equation (9), (14) and (15), the equation (13) can be rewritten as,

\[
\frac{dm(W_t)}{dt} = \begin{cases} 
  \frac{f_1(W_t)}{1 - F_1(W_t)} p_1(a + \alpha m(W_t) - m(W_t)) & \text{for } t \leq \tau \\
  \frac{f_2(W_t)}{1 - F_2(W_t)} p_2(a + \alpha m(W_t) + \alpha_2 (m(W_t) - m(W_{\tau})) - m(W_t)) & \text{for } t > \tau
\end{cases}
\]

(16)

After solving the above equations, we get the following solutions

\[
m(W_t) = \begin{cases} 
  \frac{a}{(1 - \alpha_1)} \left[1 - \left(1 - F_1(W_t)\right)^{\alpha(1 - \alpha_1)}\right] & \text{for } t \leq \tau \\
  \frac{a}{(1 - \alpha_2)} \left[1 - \left(1 - F_1(W_t)\right)^{\alpha(1 - \alpha_2)}\right] \left(\frac{1 - F_2(W_t)}{1 - F_1(W_t)}\right)^{\alpha_2(1 - \alpha_2)} & \text{for } t > \tau
\end{cases}
\]

and

\[
F_2(W_t) = 1 - \exp\left(-b_2 W_t\right) \quad \text{for } t > \tau
\]

Substituting \( F_1(W_t) \) and \( F_2(W_t) \) into Equation (16), we get:

\[
m(W_t) = \begin{cases} 
  \frac{a}{(1 - \alpha_1)} \left[1 - \exp(-b_1 W_t)\right] & \text{for } t \leq \tau \\
  \frac{a}{(1 - \alpha_2)} \left[1 - \exp(-b_1 W_t)\right] \left(\frac{1 + \beta_1 W_t}{1 + \beta_1 W_t}\right)^{\alpha_2(1 - \alpha_2)} & \text{for } t > \tau
\end{cases}
\]

(17)

The above model can be reduced to the model given by Shyur [3] if we consider the perfect debugging and no fault generation.

**SRGM-2**

Let \( F(W_t) \) be a two-stage Erlangian distribution function i.e.,

\[
F_1(W_t) = 1 - \left(1 + b_1 W_t\right) \exp\left(-b_1 W_t\right) \quad \text{for } t \leq \tau
\]

and

\[
F_2(W_t) = 1 - \left(1 + b_2 W_t\right) \exp\left(-b_2 W_t\right) \quad \text{for } t > \tau
\]

Substituting \( F_1(W_t) \) and \( F_2(W_t) \) into Equation (16), we get:

\[
m(W_t) = \begin{cases} 
  \frac{a}{(1 - \alpha_1)} \left[1 - \exp\left(-b_1 W_t\right)\right] & \text{for } t \leq \tau \\
  \frac{a}{(1 - \alpha_2)} \left[1 - \exp\left(-b_1 W_t\right)\right] \left(\frac{1 + \beta_1 W_t}{1 + \beta_1 W_t}\right)^{\alpha_2(1 - \alpha_2)} & \text{for } t > \tau
\end{cases}
\]

(18)

**SRGM-3**

Let \( F(W_t) \) be a logistic distribution function

\[
F_1(W_t) = \frac{1 - \exp\left(-b_1 W_t\right)}{(1 + \beta_1 W_t)} \quad \text{for } t \leq \tau
\]

And
\[ F_z(W_t) = \frac{1 - \exp(-b_z W_t)}{1 + \beta_z \exp(-b_z W_t)} \quad \text{for } t > \tau \]

Then the corresponding mean value function is given by:

\[
m(W_t) = \frac{a}{1-\alpha_2} \left[ \frac{(1+\beta_1 \exp(-\alpha_2 W_t)) p_1^{1-\alpha_1} \exp(-\beta_1 p_1 (1-\alpha_1) W_t)}{1-\alpha_2} \right] + \frac{\alpha_2}{1-\alpha_2} m(W_\tau) \quad \text{for } t > \tau
\]

(19)

For further simplifying the estimation procedure we may assume \( \alpha_1 = \alpha_2 = \alpha \) and \( p_1 = p_2 = p \).

**MODEL VALIDATION, COMPARISON CRITERIA AND DATA ANALYSES**

**Model Validation**
To illustrate the estimation procedure and application of the SRGM (existing as well as proposed) we have carried out the data analysis of real software data set. The parameters of the models have been estimated using statistical package SPSS and the change-point of the data sets have been judged by using change-point analyzer.

**Data set 1 (DS-1)**
The first data set (DS-1) had been collected during 35 months of testing a radar system of size 124 KLOC and 1301 faults were detected during testing. This data is cited from Brooks and Motley [19]. The change-point for this data set is 17th month.

**Data set 2 (DS-2)**
The second data set (DS-2) had been collected during 19 weeks of testing a real time command and control system and 328 faults were detected during testing. This data is cited from Ohba [7]. The change-point for this data set is 6th week.

**Comparison Criteria for SRGM**
The performance of SRGM are judged by their ability to fit the past software fault data (goodness of fit) and predicting the future behavior of the fault.

**Goodness of Fit criteria**
The term goodness of fit is used in two different contexts. In one context, it denotes the question if a sample of data came from a population with a specific distribution. In another context, it denotes the question of “How good does a mathematical model (for example a linear regression model) fit to the data”?

**The Mean Square -Error (MSE):**
The model under comparison is used to simulate the fault data, the difference between the expected values, \( \hat{m}(t_i) \) and the observed data \( y_i \) is measured by MSE as follows.

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

where \( k \) is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit [16].

**Coefficient of Multiple Determination (R²):**
We define this coefficient as the ratio of the sum of squares resulting from the trend model to that from constant model subtracted from 1.

\[ R^2 = 1 - \frac{\text{Residual SS}}{\text{Corrected SS}} \]

\( R^2 \) measures the percentage of the total variation about the mean accounted for the fitted curve. It ranges in value from 0 to 1. Small values indicate that the model does not fit the data well. The larger \( R^2 \), the better the model explains the variation in the data [16].

**Bias**
The difference between the observation and prediction of number of failures at any instant of time \( i \) is known as \( PE_i \) (prediction error). The average of \( PE_i \) is known as bias. Lower the value of Bias better is the goodness of fit [5].

**Variation**
The standard deviation of prediction error is known as variation.

\[ Variation = \sqrt{\frac{1}{N-1} \sum (PE_i - Bias)^2} \]

Lower the value of Variation better is the goodness of fit [5].

**Root Mean Square Prediction Error**
It is a measure of closeness with which a model predicts the observation.

\[ RMSPE = \sqrt{Bias^2 + Variation^2} \]
Lower the value of Root Mean Square Prediction Error better is the goodness of fit [5].

**Data Analyses**

The SRGM with mean value function \(m(W_t)\) are estimated for finding their unknown parameters. For testing effort estimation we have worked out results on all three effort functions namely Exponential, Rayleigh and Weibull (Effort function are defined in Appendix). But for model parameter estimation we have used Weibull function as it gives best results as compared to other two effort functions.

**For DS-1**

The parameter estimation and comparison criteria results for DS-1 of all the models under consideration can be viewed through Table III(a) and III(b). It is clear from the table that the value of \(R^2\) for SRGM-1 is higher and value of MSE is lower in comparison with other models and provides better goodness of fit for DS-1.

**For DS-2**

The parameter estimation and comparison criteria results for DS-2 of all the models under consideration can be viewed through Table IV(a) and IV(b). It is clear from the table that the value of \(R^2\) for SRGM-1 is higher and value of MSE is lower in comparison with other models and provides better goodness of fit for DS-2.

### Table I: Estimation of testing Effort Function Parameters for DS-1

<table>
<thead>
<tr>
<th>Testing Effort Function</th>
<th>Parameter Estimation</th>
<th>(\tilde{W})</th>
<th>(v)</th>
<th>(l)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential</td>
<td></td>
<td>1030421</td>
<td>.0000461</td>
<td>-</td>
</tr>
<tr>
<td>Rayleigh</td>
<td></td>
<td>2873</td>
<td>.00173</td>
<td>-</td>
</tr>
<tr>
<td>Weibull</td>
<td></td>
<td>2669</td>
<td>.0007729</td>
<td>2.07</td>
</tr>
</tbody>
</table>

### Table II: Estimation of testing Effort Function Parameters for DS-2

<table>
<thead>
<tr>
<th>Testing Effort Function</th>
<th>Parameter Estimation</th>
<th>(\tilde{W})</th>
<th>(v)</th>
<th>(l)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential</td>
<td></td>
<td>8544</td>
<td>.000288</td>
<td>-</td>
</tr>
<tr>
<td>Rayleigh</td>
<td></td>
<td>49</td>
<td>.013681</td>
<td>-</td>
</tr>
<tr>
<td>Weibull</td>
<td></td>
<td>799</td>
<td>.002328</td>
<td>1.11</td>
</tr>
</tbody>
</table>

### Table III(a): Model Parameter Estimation Results (DS-1)

<table>
<thead>
<tr>
<th>Models</th>
<th>(a)</th>
<th>(b_1)</th>
<th>(b_2)</th>
<th>(\beta_1)</th>
<th>(\beta_2)</th>
<th>(p)</th>
<th>(\alpha)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM-1</td>
<td>1328</td>
<td>.076</td>
<td>.146</td>
<td>-</td>
<td>-</td>
<td>.016</td>
<td>.0003</td>
</tr>
<tr>
<td>SRGM-2</td>
<td>1308</td>
<td>.079</td>
<td>.127</td>
<td>-</td>
<td>-</td>
<td>.018</td>
<td>.0229</td>
</tr>
<tr>
<td>SRGM-3</td>
<td>1400</td>
<td>.328</td>
<td>.328</td>
<td>.335</td>
<td>.999</td>
<td>.004</td>
<td>.0950</td>
</tr>
</tbody>
</table>

### Table III(b): Model Comparison Results (DS-1)

<table>
<thead>
<tr>
<th>Models</th>
<th>(R^2)</th>
<th>MSE</th>
<th>BIAS</th>
<th>VARIATION</th>
<th>RMSPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM-1</td>
<td>.9982</td>
<td>371.06</td>
<td>-2.886</td>
<td>19.32</td>
<td>19.54</td>
</tr>
<tr>
<td>SRGM-2</td>
<td>.9971</td>
<td>613.77</td>
<td>-7.442</td>
<td>23.98</td>
<td>25.10</td>
</tr>
<tr>
<td>SRGM-3</td>
<td>.9947</td>
<td>1127.1</td>
<td>-0.0003</td>
<td>34.06</td>
<td>34.06</td>
</tr>
</tbody>
</table>

### Table IV(a): Model Parameter Estimation Results (DS-2)

<table>
<thead>
<tr>
<th>Models</th>
<th>(a)</th>
<th>(b_1)</th>
<th>(b_2)</th>
<th>(\beta_1)</th>
<th>(\beta_2)</th>
<th>(p)</th>
<th>(\alpha)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM-1</td>
<td>351</td>
<td>.127</td>
<td>.193</td>
<td>-</td>
<td>-</td>
<td>.236</td>
<td>.156</td>
</tr>
<tr>
<td>SRGM-2</td>
<td>328</td>
<td>.139</td>
<td>.122</td>
<td>-</td>
<td>-</td>
<td>.553</td>
<td>.179</td>
</tr>
<tr>
<td>SRGM-3</td>
<td>375</td>
<td>.153</td>
<td>.147</td>
<td>1.16</td>
<td>1.32</td>
<td>.278</td>
<td>.137</td>
</tr>
</tbody>
</table>

### Table IV(b): Model Comparison Results (DS-2)

<table>
<thead>
<tr>
<th>Models</th>
<th>(R^2)</th>
<th>MSE</th>
<th>BIAS</th>
<th>VARIATION</th>
<th>RMSPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM-1</td>
<td>.9925</td>
<td>77.72</td>
<td>-0.42</td>
<td>9.047</td>
<td>9.057</td>
</tr>
<tr>
<td>SRGM-2</td>
<td>.9903</td>
<td>141.7</td>
<td>-2.014</td>
<td>12.05</td>
<td>12.22</td>
</tr>
<tr>
<td>SRGM-3</td>
<td>.9908</td>
<td>94.79</td>
<td>-0.003</td>
<td>10.00</td>
<td>10.00</td>
</tr>
</tbody>
</table>

### GOODNESS OF FIT CURVES

For DS-1

![Goodness of fit curves](chart.png)
CONCLUSION

In this paper we have developed a unified framework for testing effort dependent software reliability growth models incorporating change-point concept with two types of imperfect debugging. The framework presented here proves to be excellent for deriving a wide variety of effort dependent models by using different probability distribution functions. The technique is simple and presents a unique methodology for developing many new as well as existing models for different design environment. With this approach, we can derive existing models and propose new model. In this paper we have restricted ourselves to the standard distributions e.g. Exponential, Weibull and Erlang k-type for correction times. Their validity and accuracy have been carried out on two real software failure datasets. The results obtained are quite encouraging as can be viewed through the numerical illustrations shown in tables obtained after the parameter estimation.

FUTURE SCOPE

In future work we are working upon the possibility of including some new distribution functions like Normal, Gamma etc. for correction times. Their capability to represent the severity and delays in the fault correction need to be numerically worked out and checked. The concept of unification provides an area of interesting study which can ease out the problem of model selection for the software developer and thus make these techniques more accessible and applicable.

REFERENCE


Continued on Page No. 522
Unified Framework For Developing Testing Effort Dependent Software Reliability Growth Models With Change Point and Imperfect Debugging

Continued from Page No. 516


APPENDIX
MODELING TESTING EFFORT

The testing resources spent during testing of any software basically, include manpower used for fault detection / removal and CPU time spent in executing software under test. Greater the amount of testing effort faster is the testing process. The testing effort (resources) that govern the pace of testing for almost all the software projects are [4]:

1. Manpower
2. Computer time.

The key function of manpower engaged in software testing is to design and run test cases and compare the test results with desired specifications. Any departure from the specifications is termed as a failure. On a failure the fault causing it is identified and then removed by failure correction personnel. During testing continuous monitoring is done to analyze the progress of testing and quality achieved. The computer facilities represent the computer time, which is necessary for failure identification and correction.

The Functions which will be used in this paper to explain the testing effort are- Exponential, Rayleigh and Weibull.

They can be derived from the assumption that, "The testing effort rate is proportional to the testing resources available".

\[
\frac{dW_t}{dt} = v(t) \left[ \alpha - W_t \right]
\]

Where \( v(t) \) is the time dependent rate at which testing resources are consumed, with respect to remaining available resources.

Case 1: When \( v(t)=v, \) a constant, we get Exponential function:

\[
W_t = \alpha \left( 1 - e^{-vt} \right)
\]

Case 2: If \( v(t)=vt, \) we get Rayleigh type curve:

\[
W_t = \alpha \left( 1 - e^{-vt^2/2} \right)
\]

Case 3: If \( v(t)=v.t^{1/2}, \) we get Weibull function:

\[
W_t = \alpha \left( 1 - e^{-v.t^{1/2}} \right)
\]

To study the testing effort process, one of the above functions can be selected.