Software Reliability Growth Modeling For Distributed Environment Using Component-Specific Testing-Effort Functions

P. K. Kapur¹ Kalpana Yadav² Avneesh Chauhan³ Prashant Johri³

¹Department of Operational Research, University of Delhi, Delhi, pkkapur1@gmail.com
²I. G. I. T., Guru Gobind Singh Indraprastha University, Delhi, kyadav11@yahoo.com
³Integrated Academy of Management & Technology, UP Technical University, johri.prashant@gmail.com

ABSTRACT
Software’s with their components developed in distributed environment shows an astounded trend as they have high speed and elevated reliability performance. Fault removal phenomenon for reused and newly developed components in distributed environment with component-specific testing-effort function is illustrated in the paper. Explicit expressions for the mean number of individual types of faults are obtained by considering N-types of faults for newly developed component and single type of faults for reused components. Software system components developed in distributed environment consider different testing-effort based functions. For such systems, testing-effort needs to be modeled separately for each component. The overall effort of the system must then be determined from the component-specific testing-effort functions. This approach partitions the testing-effort with growth curves of varying nature among different components of the same software developed in distributed environment. To validate the analytical results of the proposed framework, numerical illustrations are provided.

KEYWORDS
Testing-effort function (TEF), Distributed development environment (DDE), Non Homogeneous Poisson Process (NHPP).

INTRODUCTION
In real time software systems, the detected errors should be removed immediately, yet a few errors may be introduced during debugging. Newly developed software should be tested to eliminate such errors before its release to the user. Software containing errors incurs high failure costs when released in market. Hence, there is a need to develop more reliable/failure-free software systems.

Software reliability has been often studied in terms of software reliability growth models (SRGM), based on observed software error data during the software testing phase. Software reliability growth models are concerned with the relation between the cumulative number of errors detected by software testing and the time span of the software testing. Software reliability growth models can estimate the expected initial error content of a software system, the expected number of remaining errors at an arbitrary testing time point, the software reliability, and so on. Several software reliability growth models have been proposed and investigated [1, 2, 3, and 4]. Software’s developed in DDE are characterized by enhanced availability and increased reliability. Distributed systems often involve development/testing teams that are situated across company sites, organizations, sectors and nations. Thus, there are special risk factors involved in comparison to the normal risks of software development. A software project developed with some or all of its components generated by different team’s present multifarious issues of quality and reliability of the software. It is necessitated to estimate, risk assess, plan and manage the development of these distributed components and the final full system release.

The models developed in DDE assume that the software system consists of a finite number of reused and newly developed software components [5, 7, and 8]. The reused components are basically taken from the software’s which are already released in the market. The probability of finding errors in such components is quite less and such errors are not fatal or critical from developers or users point of view. Hence, the reused components reveal exponential growth curves to demonstrate the debugging-lag/ delay-effect factor between failure observation and its subsequent fault/error removal is negligible. The newly developed components may fail because of the severe/critical errors as compared to the errors in reused component lying inherent in the code, and are represented by the S-shaped growth curves of varying nature to express the fact that there is a time-delay between failure observation and its corresponding fault/error removal. Fault removal phenomenon for reused and newly developed components has been modeled separately and is summed up to get the total fault removal phenomenon of the software system.

Many resources are consumed in developing a software project. It is assumed that the consumption rate of testing resource expenditures during testing phase is a constant or even do not consider such testing-effort. In reality software reliability models should be developed by incorporating different testing-effort functions. Yamada et al. [6], Musa [2] and Kapur et al. [3, 7] proposed several software reliability growth models which describe the relationship among the calendar testing, the amount of testing-effort, and the number of software errors detected.
In general, appreciable testing resources are spent on software testing in software development. The consumption curve of testing resources over the testing period can be thought of as a testing-effort curve. Testing-effort is measured by: the number of executed test cases, the amount of man-power, the CPU time spent during the testing phase, and so on. However existing software reliability growth models do not consider such testing-effort, that is, they assume that testing-effort is constant over the testing period. We should consider the effect of testing-effort on software reliability growth in order to develop more realistic software reliability growth models.

The testing-effort models prevalent in software engineering literature consider that the various components of single software developed in DDE are homogeneous entity from the viewpoint of testing-effort consumption growth curves. In reality, these components being heterogeneous entity and taken from different sources need to employ varying amount of testing resources. For such systems, testing-effort needs to be modeled separately for each component. The overall effort of the system must then be determined from the component-level efforts, within the context of the system architecture [9]. In proposing SRGM, we have taken into account the role of learning process during the testing phase by accounting for the experience gained with the progress of software testing.

In this paper, we discuss component-specific testing-effort function based software reliability growth models for DDE. The proposed models assume that the software system consists of one reused and two newly developed components. The fault removal phenomena for reused and newly developed components have been modeled separately using different testing-effort functions and are summed to obtain the total fault removal phenomenon of the software system. Finally, a goodness-of-fit comparison between proposed models and existing continuous-state space SRGM developed in distributed environment has been performed.

### Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m(X(t)))</td>
<td>mean number of faults removed in time interval ((0,t])</td>
</tr>
<tr>
<td>(m_i(X(t)), (i=1,2,3))</td>
<td>mean number of faults removed from reused and newly developed components by time (t)</td>
</tr>
<tr>
<td>(a)</td>
<td>total fault content lying inherent in code when testing starts</td>
</tr>
<tr>
<td>(b_i (i=1,2,3))</td>
<td>constant representing failure rate/fault isolation/fault removal rate per fault in reused and newly developed components</td>
</tr>
<tr>
<td>(p_i (i=1,2,3))</td>
<td>proportion of faults in reused and newly developed components, (\sum_{i=1,3} p_i = 1)</td>
</tr>
<tr>
<td>(x(t))</td>
<td>testing-effort intensity</td>
</tr>
<tr>
<td>(X(t) = \int_0^t x(y)dy)</td>
<td>cumulative testing-effort in time interval ((0,t])</td>
</tr>
<tr>
<td>(X_i(t), (i=1,2,3))</td>
<td>cumulative testing-effort consumed by time (t) for reused and newly developed components</td>
</tr>
<tr>
<td>(E)</td>
<td>total amount of testing-effort consumption</td>
</tr>
<tr>
<td>(c)</td>
<td>proportion of testing-effort consumption by reused and newly developed components, (\sum_{i=1,3} q_i = 1)</td>
</tr>
<tr>
<td>(k, k_1, k_2)</td>
<td>rate of testing-effort consumption</td>
</tr>
<tr>
<td>(f)</td>
<td>rate of consumption of testing-effort in reused, newly developed components</td>
</tr>
<tr>
<td>(p_i, (i=1,2,3))</td>
<td>shape parameters</td>
</tr>
<tr>
<td>(c_1, c_2/c_3)</td>
<td>constant representing learning parameter in logistic function of newly developed components</td>
</tr>
<tr>
<td>(f)</td>
<td>constant parameter in logistic TEF</td>
</tr>
</tbody>
</table>

### 2. PROPOSED MODELING FRAMEWORK

In this section, we will begin with basic model assumptions. Then, different testing-effort functions used in proposing SRGM are discussed. Finally, models with varied testing-effort in component-specific modules of distributed environment are proposed.

#### 2.1. Model Assumptions

The proposed models are based on following assumptions:

- Failure observation/fault removal phenomenon is modeled by an NHPP. The fault removal process i.e., the debugging process is perfect.
- Software system consists of one reused and two newly developed components. The fault removal process of reused component follows exponential curve whereas that of newly developed components follow S-shaped curve.
- Each time a failure is observed, an immediate (delayed) effort takes place to decide the cause of failure in order to remove it. The time-delay between the failure observation and its subsequent removal is assumed to represent the severity of faults. The more severe the fault, more the time delay.
- The fault isolation/removal rate with respect to testing effort intensity is proportional to the number of observed failures whose cause is yet to be identified.

#### 2.2. A General Description of Testing-Effort Functions

The testing-effort, such as, the number of executed test-cases, CPU hours expended in testing phase, etc., should be taken into account while framing an SRGM to describe the time dependent behavior of testing-effort expenditures in testing phase.
Yamada et al. [10] found that the TEF could be described by a Weibull-type distribution with the following three cases.

**Case 1 (Exponential curve):** The cumulative testing-effort consumed in $(0, t]$ is:

$$X(t) = E(1 - e^{-ct})$$  

(1)

The exponential curve is used for processes that decline monotonically to an asymptote.

**Case 2 (Rayleigh curve):** The Rayleigh curve often predicts the costs and schedules of software development well. It is frequently employed as an alternative to the exponential curve. The cumulative testing-effort consumed is:

$$X(t) = E(1 - e^{-ct^2/2})$$  

(2)

**Case 3 (Weibull curve):** The tail of the Weibull curve probability density function approaches zero asymptotically, but never reaches it. The cumulative testing-effort consumed is:

$$X(t) = E(1 - e^{-ct^k})$$  

(3)

Exponential and Rayleigh curves become special cases of the Weibull curve for $k=1$ and $k=2$ respectively.

**Case 4 (Exponentiated Weibull curve):** It is an extension of the Weibull family distributions. The total testing-effort consumption is given by:

$$X(t) = E\left(1 - e^{(-ct)^{1/k}}\right)^{k_2}$$  

(4)

**Case 5 (Logistic curve):** This TEF can be used instead of the Weibull-type curve to describe the test-effort patterns during the software development process. Logistic TEF was originally proposed by F. N. Parr [11]. It exhibits similar behavior to the Rayleigh curve, except during the early part of the project.

$$X(t) = E\left(1 + fe^{-ct}\right)^{k_3}$$  

Where $X(0) = E\left(1 + f\right)$

(5)

It can be noted that $X(0)$ is equal to zero for all the above specified testing-effort functions discussed in cases 1-4, except for logistic function defined in case 5.

### 2.3. Framework for Proposing SRGM for DDE using Component-Specific TEF

In modeling SRGM for DDE, its components are developed and tested at different geographical sites, integrated later on to produce the final product and then again tested for errors. Some of the components comprise of the library functions of existing software’s and form reused components while others are newly developed components. The testing-effort consumption curves used in these distinct components (reused and newly developed) of software up till now are of the same nature. But in reality, testing-effort applied on reusing existing components and developing entirely new components is different and has to be modeled separately. In this paper, we propose two new frameworks where the testing effort for reused and newly developed components is modeled separately. We have used exponential testing-effort function for modeling reliability growth curves of reused components as the testing team has already acquired expertise with debugging of such components resulting in lesser consumption of testing-effort. Rayleigh/Weibull/Exponential-weibull/logistic testing-effort functions are used for modeling the reliability growth curves of newly developed components as the testing team will gain an insight into the internal structure/code of such components with the passage of time. Hence, the testing-effort consumption requirement for debugging of newly developed components is more as compared to its consumption in reused components of software.

#### 2.3.1. Modeling Testing-Effort of Reused and Newly Developed Components

In this paper, we have used the exponential curve given by equation (1) for modeling the reused components of software and it can be rewritten here as:

$$X_1(t) = Eq_1\left(1 - e^{-ct}\right)$$  

(6)

For debugging of two newly developed components of software, testing-effort consumption i.e. $X_2(t)$ or $X_3(t)$, can be modeled using Rayleigh/Weibull/Exponential-weibull/logistic functions defined in equations (2), (3), (4) and (5). Hence, these equations can be rewritten as:

$$X_2(t) / X_3(t) = \begin{cases} \frac{Eq_{23} \left(1 - e^{-ct/2}\right)^{k_2}}{Eq_{23} \left(1 - e^{-ct/2}\right)^{k_2}} / \frac{Eq_{23} \left(1 - e^{-ct/2}\right)^{k_2}}{Eq_{23} \left(1 + fe^{-ct}\right)^{k_3}} \end{cases}$$  

(7)

#### 2.3.2. Modeling Testing-Effort of Final Software Product

The total testing-effort at time $t$ for the complete software system can be modeled as:

$$X(t) = X_1(t) + X_2(t) + X_3(t)$$  

(8)

Initially, modeling and estimation of unknown parameters of testing-effort for final software product is done using equation (8) on complete data set. Then, using estimated parameter
results, component-specific testing-effort is calculated. Now, the component-specific testing-effort is designated as \( X_1(t) \) or \( X_2(t)/X_3(t) \) by looking at the nature of different component-specific testing-effort curves. Modeling of reliability for complete software built in distributed environment is to be done using \( X_1(t) \), \( X_2(t) \) and \( X_3(t) \) for different components instead of using \( X(t) \) for all components as used previously in the literature.

### 2.3.3. Modeling of Faults in Software

For modeling of faults in the final software product using DDE, we need to study and model the faults prevalent in its reused and newly developed components. To propose a classification scheme based on the types of component, Kapur et al. [7] assumed that reused components have already undergone considerable testing and should have lower fault content than those recently developed. In proposing our models, we are taking into consideration only one reused and two newly developed components. The proposed models assume that the testing phase consists of three processes namely: failure observation, fault detection/isolation and fault removal. The software faults are categorized into three types according to the amount of testing effort needed to remove them. The time delay between failure observation, subsequent fault detection/isolation and its removal is assumed to represent the testing effort.

Here, \( X_{1/2/3}(t) \) is the amount of testing-effort expended by arbitrary testing time \( t \). For derivation purposes, \( X_{1/2/3}(0) \neq 0 \) and \( X_{1/2/3}^*(t)=X_{1/2/3}(t)-X_{1/2/3}(0) \) is taken into consideration. It can be noted that \( X_{1/2/3}(0) \) is equal to zero for all the above specified testing effort function except for logistic function as described in case (5) of TEFs.

#### 2.3.3.1. Modeling of Faults in Reused Component

The debugging process of reused components does not require much skill and experience with the testing team. As faults in reused components are simple, time delay between the failure observation, fault detection/removal is negligible.

For simple faults, the fault removal phenomenon is modeled by (extension of [6]):

\[
m_1(X_1(t)) = \begin{cases} 
  ap_1 \left( 1 - e^{-b_1X_1(t)} \right); & \text{for } X_1(0) = 0 \\
  ap_1 \left( 1 - e^{-b_1X_1^*(t)} \right); & \text{for } X_1(0) \neq 0 
\end{cases}
\]  

(9)

#### 2.3.3.2. Modeling of Faults in Newly Developed Components

The debugging process of newly developed components consumes more testing-effort as compared to the debugging process of reused components of software as the nature of faults existing in them is hard/complex. This means that the testing team will have to spend more time to analyze the cause of failure and therefore requires greater efforts to remove them. Hence the removal process for such faults is modeled as a two stage process. The first stage describes the failure observation process with constant failure observation rate. The second stage of a two-stage process describes the delayed fault detection/removal process in which the detection/removal rate can be assumed to be either a non-time dependent (constant) or a time dependent function.

If in second stage, the fault detection/removal rate is assumed to be a non-time dependent (constant) factor, then its mean value function will be given by (extension of [3]):

\[
m_2(X_2(t)) = \begin{cases} 
  ap_2 \left[ 1 - (1 + b_2X_2(t))e^{-b_2X_2(t)} \right]; & \text{for } X_2(0) = 0 \\
  ap_2 \left[ 1 - (1 + b_2X_2(t))e^{-b_2X_2^*(t)} \right]; & \text{for } X_2(0) \neq 0 
\end{cases}
\]  

(10)

Now, let us consider the fault detection/removal rate to be a time dependent function. The reason for this assumption is to incorporate the effect of learning on the removal process. With each fault removal insight is gained into the nature of faults present and function described called logistic function can account for that. So its mean value function will be given by (extension of [12]):

\[
m_2(X_2(t)) = \begin{cases} 
  ap_2 \left[ \frac{1 + b_2X_2(t)}{1 + b_2 e^{-b_2X_2(0)}} \right] e^{-b_2X_2(t)}; & \text{for } X_2(0) = 0 \\
  ap_2 \left[ \frac{1 + b_2X_2(t)}{1 + b_2 e^{-b_2X_2(0)}} \right] e^{-b_2X_2^*(t)}; & \text{for } X_2(0) \neq 0 
\end{cases}
\]  

(11)

There can be components still having harder faults or complex faults. These faults require more effort for removal after their detection/isolation. Hence they need to be modeled with greater time lag between failure observation and removal. The first stage describes the failure observation process with constant failure observation rate, the second stage describes the fault detection/isolation process with constant detection/isolation rate and the third stage describes fault removal process. The third stage of a three-stage process describes the delayed fault removal process after its detection/isolation in which the removal rate can be assumed to be either a non-time dependent (constant) or a time dependent function.

If in third stage, the fault removal rate is assumed to be a non-time dependent (constant) factor, then its mean value function will be given by (extension of [3]):
2.3.4. Proposed SRGM for DDE using Component-Specific TEF

The total removal phenomenon is modeled by the superposition of three mean value functions given by \( m_1(X_1(t)) \), \( m_2(X_2(t)) \) and \( m_3(X_3(t)) \), i.e.:

\[
m(X(t)) = m_1(X_1(t)) + m_2(X_2(t)) + m_3(X_3(t))
\]

Here, we will be incorporating only one reused (simple faults/one stage) and two newly developed (hard faults/two stage, complex faults/three stage) components in our proposed SRGM.

\[
m_3(X_3(t)) = \begin{cases} 
  ap_3 \left[ 1 - \left( 1 + b_3 X_3(t) + \frac{b_3^2 (X_3(t))^2}{2} \right) e^{-b_3 X_3(t)} \right] ; \\
  \text{for } X_3(0) = 0 \\
  \end{cases}
\]

\[
m_3(X_3(t)) = \begin{cases} 
  ap_3 \left[ 1 - \left( 1 + b_3 X_3(t) + \frac{b_3^2 (X_3(t))^2}{2} \right) e^{-b_3 \tau X_3^*(t)} \right] ; \\
  \text{for } X_3(0) \neq 0 \\
  \end{cases}
\]

If we consider the fault detection/removal rate to be a time dependent function, then its mean value function will be given by (extension of [12]):

\[
m_3(X_3(t)) = \begin{cases} 
  ap_3 \left[ 1 - \left( 1 + b_3 X_3(t) + \frac{b_3^2 (X_3(t))^2}{2} \right) e^{-b_3 X_3(t)} \right] ; \\
  \text{for } X_3(0) = 0 \\
  \end{cases}
\]

\[
m_3(X_3(t)) = \begin{cases} 
  ap_3 \left[ 1 - \left( 1 + b_3 X_3(t) + \frac{b_3^2 (X_3(t))^2}{2} \right) e^{-b_3 \tau X_3^*(t)} \right] ; \\
  \text{for } X_3(0) \neq 0 \\
  \end{cases}
\]

Proposed SRGM-1

Using equations (9), (10) and (12), we get (considering only the cases where the fault removal rate in nth-stage of an n-stage process is a constant factor). Pl. refer the model as No. (15) presented above.

Now, we can have different cases for proposed SRGM-1 with exponential curve of \( X_1(t) \) and rayleigh/weibull/Exponential-weibull/logistic curves of \( X_2(t) \) or \( X_3(t) \).

Proposed SRGM-2

Using equations (9), (11) and (13), we get (considering only the cases where the fault removal rate in nth-stage of an n-stage process is a time-dependent factor, keeping constant removal rate in simple faults):
3. Model Validation and Comparison Criteria

3.1. Model Validation

To assess the performance of proposed SRGM, we have carried out the parameter estimation on two real software failure datasets.

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 [14].

Data set 2 (DS-2)

This data is cited from Wood [13]. The software was tested for 20 weeks during which 10000 computer hours were used and 100 faults were removed.

Comparison Criteria: Goodness of Fit Criteria

In general, a model can be analyzed according to its ability to reproduce the observed behavior of the software. Some of the comparison criteria are:

3.2.1. The Mean Square Fitting Error (MSE) [3]: Using the estimated values of the parameters, the estimated values for failure data are calculated. The difference between the estimated values, $m(t_i)$ and the observed values $y_i$ is measured by MSE as follows:

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

(17)

Where $k$ is the number of observations. Lower value of MSE indicates less fitting error, thus better goodness of fit.

3.2.2. Coefficient of Multiple Determination ($R^2$) [3]: This measure can be used to investigate whether a significant trend exists in the observed failure intensity. This coefficient is defined as the ratio of the Sum of Squares (SS) resulting from the trend model to that from a constant model subtracted from 1, that is:

$$
R^2 = 1 - \frac{\text{residual SS}}{\text{corrected SS}}
$$

(18)

$R^2$ measures the percentage of the total variation about the mean accounted for by the fitted curve. It ranges in value from 0 to 1. Small values indicate that the model does not fit the data well.

3.2.3. Bias [15]: 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 PEs is known as bias. Lower the value of Bias better is the goodness of fit.

3.2.4. Variation [15]: The standard deviation of PE is known as variation.

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

(19)

Lower the value of Variation better is the goodness of fit.

3.2.5. Root Mean Square Prediction Error [15]: It is a measure of closeness with which a model predicts the observation.

$$
RMSPE = \sqrt{\text{Bias}^2 + \text{Variation}^2}
$$

(20)

Lower the value of Root Mean Square Prediction Error better is the goodness of fit.

Data Analysis and Model Comparison

While checking the accuracy of different cases of two proposed SRGM with respect to component-specific testing-effort functions developed in DDE, we have first estimated the unknown parameters of testing-effort for final software product using equation (8) on effort cumulative consumption data. Then, using estimated parameter results, component-specific testing-effort is calculated. Now, the component-specific testing-effort is designated as $X_1(t)$ or $X_2(t)/X_3(t)$ by looking at the nature of different component-specific testing-effort curves. Modeling of reliability for complete software built in distributed environment is to be done using $X_1(t)$, $X_2(t)$ and $X_3(t)$ for different components instead of using $X(t)$ for all components as used previously in the literature.
To judge the fitting of various cases of two proposed SRGM given by equations (15) and (16), we have compared the results to SRGM given by Yamada et al. [5] with Exponential, Rayleigh, Weibull, Exponential-Weibull, logistic testing-effort functions respectively, and used MSE, $R^2$, bias, variation and RMSPE as the performance measures.

3.2. First Application (using DS-1)

The estimated values of parameters for different effort functions are provided in Table-1. Once the estimation for the effort function is over, parameters of mean value function $m(t)$ given by equations (15) and (16) are worked out. Table-2 gives the estimated values for the parameters while Table-3 provides the comparison criteria results of the models by Yamada et al. [5] and those of the proposed models.

Table-1: Estimation of Testing Effort Function Parameters

Now we will estimate the model parameters using different effort functions.

Table-2: Model Parameters Estimation Results

Table-3: Model Comparison Criteria Results

3.3. Second Application (using DS-2)

Testing effort data of software collected in the form of testing effort consumed in some time period is collected and then the unknown parameters in the testing effort functions are estimated. Using these estimated parameters values, we estimate the parameters in the proposed model given by equations (30) and (31). The estimated values of parameters for the different effort function are provided in Table-4. Table-5 gives the estimated values for the parameters while Table-6 provides the comparison criteria results.

Table-4: Estimation of Testing Effort Function Parameters

Now we will estimate the model parameters using different effort functions.

Table-5: Model Parameters Estimation Results

Table-6: Model Comparison Criteria Results

From Table-6, it can be easily observed that different cases of proposed SRGM-1 and SRGM-2 provide best fit than the existing model given by Yamada et al. [5] in terms of $R^2$, MSE, bias, variation and RMSPE for different effort functions.

4. CONCLUSION

To develop software system in distributed development environment, its different components are designed and tested at various sites. A majority of testing-effort models consider the software system produced in DDE as a single, homogeneous entity. Modern software systems, however, are heterogeneous and may be assembled using components from different sources. For such systems, testing-effort needs to be modeled separately for each component of software. Here, we have proposed software reliability growth models for DDE using component-specific testing-effort functions assuming that the software system consists of a one reused and two newly developed components. However, these models can be extended for a finite ‘n’ number of reused and finite ‘m’ number of newly developed components.

FUTURE SCOPE

A number of other factors like imperfect debugging, error generation, etc. can be incorporated in the proposed framework. This proposal of models has, in fact, paved way for future research in other interdisciplinary fields of software reliability.

REFERENCES


Table-1: Estimation of Testing Effort Function Parameters for DS-1

<table>
<thead>
<tr>
<th>Models</th>
<th>E</th>
<th>c</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential</td>
<td>.35236</td>
<td>.0297</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rayleigh</td>
<td>.10153</td>
<td>.0406</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weibull</td>
<td>.57046</td>
<td>.0139</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential-weibull</td>
<td>.30245</td>
<td>.00201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistic</td>
<td>.1715</td>
<td>.38487</td>
<td>.3713</td>
<td>.7764</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential+Rayleigh+Weibull</td>
<td>.13981</td>
<td>.0693</td>
<td>.2060</td>
<td>.1266</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential+Rayleigh+Exponential-weibull</td>
<td>.14415</td>
<td>.0683</td>
<td>.2359</td>
<td>.0067</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential+Weibull+Logistic</td>
<td>.14764</td>
<td>.6143</td>
<td>.0151</td>
<td>.3713</td>
<td>.7764</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Models</th>
<th>k</th>
<th>K1</th>
<th>K2</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>R²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential</td>
<td>.9985</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rayleigh</td>
<td>.9550</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weibull</td>
<td>.9550</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential-weibull</td>
<td>.9986</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistic</td>
<td>.9970</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential+Rayleigh+Weibull</td>
<td>.9512</td>
<td>.4179</td>
<td>.3014</td>
<td>.2804</td>
<td>.9995</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential+Rayleigh+Exponential-weibull</td>
<td>5.4954</td>
<td>2.6808</td>
<td>.1672</td>
<td>.2364</td>
<td>.9962</td>
<td>.9996</td>
<td></td>
</tr>
<tr>
<td>Exponential+Weibull+Logistic</td>
<td>1.6629</td>
<td>.1037</td>
<td>.7295</td>
<td>.1672</td>
<td>.9996</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graph for DS-1
Table-2: Model Parameters Estimation Results for DS-1

<table>
<thead>
<tr>
<th>Models</th>
<th>Effort function</th>
<th>A</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kapur et al. [9]</td>
<td>Exponential</td>
<td>2364</td>
<td>.0174</td>
<td>.0005</td>
<td>.0006</td>
</tr>
<tr>
<td></td>
<td>Rayleigh</td>
<td>2322</td>
<td>.0198</td>
<td>.0005</td>
<td>.0006</td>
</tr>
<tr>
<td></td>
<td>Weibull</td>
<td>2375</td>
<td>.0305</td>
<td>.0005</td>
<td>.0005</td>
</tr>
<tr>
<td></td>
<td>Exponential-weibull</td>
<td>2346</td>
<td>.0055</td>
<td>.0010</td>
<td>.0015</td>
</tr>
<tr>
<td></td>
<td>Logistic</td>
<td>2328</td>
<td>.0221</td>
<td>.0005</td>
<td>.0004</td>
</tr>
<tr>
<td>Proposed SRGM-1</td>
<td>Exponential+Rayleigh+Weibull</td>
<td>63017</td>
<td>.00001</td>
<td>.0001</td>
<td>.0013</td>
</tr>
<tr>
<td>Proposed SRGM-1</td>
<td>Exponential+Rayleigh+Exponential-weibull</td>
<td>50160</td>
<td>.00001</td>
<td>.0006</td>
<td>.0003</td>
</tr>
<tr>
<td>Proposed SRGM-1</td>
<td>Exponential+Weibull+Logistic</td>
<td>2694</td>
<td>.1210</td>
<td>.0007</td>
<td>.0001</td>
</tr>
<tr>
<td>Proposed SRGM-2</td>
<td>Exponential+Rayleigh+Weibull</td>
<td>62921</td>
<td>.00001</td>
<td>.0001</td>
<td>.0100</td>
</tr>
<tr>
<td>Proposed SRGM-2</td>
<td>Exponential+Rayleigh+Exponential-weibull</td>
<td>24349</td>
<td>.00001</td>
<td>.0012</td>
<td>.0074</td>
</tr>
<tr>
<td>Proposed SRGM-2</td>
<td>Exponential+Weibull+Logistic</td>
<td>5835</td>
<td>.0002</td>
<td>.0005</td>
<td>.3210</td>
</tr>
</tbody>
</table>

Table-3: Model Comparison Criteria Results for DS-1

<table>
<thead>
<tr>
<th>Models</th>
<th>Effort function</th>
<th>R²</th>
<th>MSE</th>
<th>BIAS</th>
<th>VARIATION</th>
<th>RMSPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kapur et al. [9]</td>
<td>Exponential</td>
<td>.9379</td>
<td>19229</td>
<td>36.775</td>
<td>135.44</td>
<td>140.35</td>
</tr>
<tr>
<td></td>
<td>Rayleigh</td>
<td>.9149</td>
<td>26342</td>
<td>36.532</td>
<td>160.87</td>
<td>164.96</td>
</tr>
<tr>
<td></td>
<td>Weibull</td>
<td>.9405</td>
<td>18435</td>
<td>30.955</td>
<td>134.44</td>
<td>135.96</td>
</tr>
<tr>
<td></td>
<td>Exponential-weibull</td>
<td>.9345</td>
<td>20281</td>
<td>43.828</td>
<td>136.50</td>
<td>143.37</td>
</tr>
<tr>
<td></td>
<td>Logistic</td>
<td>.9706</td>
<td>9125</td>
<td>1.5963</td>
<td>137.62</td>
<td>137.63</td>
</tr>
<tr>
<td>Proposed SRGM-1</td>
<td>Exponential+Rayleigh+Weibull</td>
<td>.9420</td>
<td>2759</td>
<td>12.754</td>
<td>51.748</td>
<td>53.297</td>
</tr>
<tr>
<td>Proposed SRGM-1</td>
<td>Exponential+Rayleigh+Exponential-weibull</td>
<td>.9350</td>
<td>7731</td>
<td>.00495</td>
<td>90.338</td>
<td>90.338</td>
</tr>
<tr>
<td>Proposed SRGM-1</td>
<td>Exponential+Weibull+Logistic</td>
<td>.9393</td>
<td>8058</td>
<td>16.588</td>
<td>90.046</td>
<td>91.562</td>
</tr>
<tr>
<td>Proposed SRGM-2</td>
<td>Exponential+Rayleigh+Weibull</td>
<td>.9911</td>
<td>2759</td>
<td>12.754</td>
<td>51.748</td>
<td>53.297</td>
</tr>
<tr>
<td>Proposed SRGM-2</td>
<td>Exponential+Rayleigh+Exponential-weibull</td>
<td>.9948</td>
<td>1609</td>
<td>1.0532</td>
<td>41.197</td>
<td>41.211</td>
</tr>
</tbody>
</table>

Table-4: Estimation of Testing Effort Function Parameters for DS-2

<table>
<thead>
<tr>
<th>Models</th>
<th>E</th>
<th>c</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential</td>
<td>3608.14</td>
<td>.0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rayleigh</td>
<td>2873</td>
<td>.0019</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weibull</td>
<td>2669</td>
<td>.0007</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential-weibull</td>
<td>3815</td>
<td>.1624</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistic</td>
<td>2666</td>
<td>.1815</td>
<td>38.6434</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential+Rayleigh</td>
<td>7805</td>
<td>.00001</td>
<td>.0017</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential+Exponential-weibull</td>
<td>1314</td>
<td>.0179</td>
<td>.0175</td>
<td>1.2810</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential+Weibull+Logistic</td>
<td>2669</td>
<td>.0042</td>
<td>.0007</td>
<td>.0620</td>
<td>5.2099</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Proceedings of the 4th National Conference; INDIACom-2010

Models | Effort function | a | b₁ | b₂ | b₃
--- | --- | --- | --- | --- | ---
Rayleigh | 1472.2381 | .0022 | .0027 | .00001
Weibull | 1463.4328 | .0023 | .0027 | .00001
Exponential - weibull | 1463.4328 | .0024 | .0028 | .00001
Logistic | 1463.6984 | .0022 | .0028 | .00001

Proposed SRGM-1 | Exponential+ Rayleigh+ Weibull | 1333.1836 | .9856 | .0029 | .00001
Proposed SRGM-1 | Exponential+ Rayleigh+ Exponential- weibull | 7443.3018 | .0083 | .0948 | .00001
Proposed SRGM-2 | Exponential+ Rayleigh+ Weibull | 2773.9263 | .0865 | .0161 | .00001
Proposed SRGM-2 | Exponential+ Rayleigh+ Weibull | 1343.5848 | .0038 | .0229 | .00001
Proposed SRGM-2 | Exponential+ Rayleigh+ Exponential- weibull | 1458.4831 | .0016 | .0440 | .00003
Proposed SRGM-2 | Exponential+ Weibull+ Logistic | 1328.6601 | .8524 | .00390 | .0165

Table-5: Model Parameters Estimation Results for DS-2

<table>
<thead>
<tr>
<th>p₁</th>
<th>p₂</th>
<th>R²</th>
<th>MSE</th>
<th>BIAS</th>
<th>VARIATION</th>
<th>RMS PE</th>
</tr>
</thead>
<tbody>
<tr>
<td>.0152</td>
<td>.457</td>
<td>.9679</td>
<td>.0188</td>
<td>.0865</td>
<td>8.8263</td>
<td>5712.16</td>
</tr>
<tr>
<td>.0018</td>
<td>.977</td>
<td>.0104</td>
<td>.0962</td>
<td>.9278</td>
<td>3.3990</td>
<td>73.35</td>
</tr>
<tr>
<td>.0003</td>
<td>.9575</td>
<td>.0624</td>
<td>.9872</td>
<td>.3323</td>
<td>2.3052</td>
<td></td>
</tr>
<tr>
<td>.0003</td>
<td>.9575</td>
<td>.0624</td>
<td>.9872</td>
<td>.3323</td>
<td>2.3052</td>
<td></td>
</tr>
<tr>
<td>.0136</td>
<td>.9764</td>
<td>.8117</td>
<td>.9792</td>
<td>.0080</td>
<td>0.0796</td>
<td></td>
</tr>
<tr>
<td>.0684</td>
<td>.9336</td>
<td>.0064</td>
<td>.00001</td>
<td>.9416</td>
<td>15.862</td>
<td>77.46</td>
</tr>
<tr>
<td>.9999</td>
<td>.4644</td>
<td>.0375</td>
<td>.9810</td>
<td>3.4573</td>
<td>1.15</td>
<td></td>
</tr>
<tr>
<td>.1614</td>
<td>.8388</td>
<td>.00001</td>
<td>.1475</td>
<td>.7706</td>
<td>.9990</td>
<td>201.28</td>
</tr>
<tr>
<td>.9799</td>
<td>.00001</td>
<td>.8262</td>
<td>.0006</td>
<td>.9388</td>
<td>3.4573</td>
<td>1.15</td>
</tr>
<tr>
<td>.0680</td>
<td>.8484</td>
<td>.0834</td>
<td>.8592</td>
<td>.3968</td>
<td>.9917</td>
<td>2.3807</td>
</tr>
</tbody>
</table>

Table-6

Graph for DS-2