A New Approach For Developing Testing Effort Dependent Software Reliability Growth Models

Anu G. Aggarwal  Ravi Kumar  P.K. Kapur
Department of Operational Research, University of Delhi, India, pkkapur1@gmail.com

ABSTRACT
Several software reliability growth models have been presented in the literature in the last three decades. They have been developed for uniform and non-uniform operational profile. Some of them are flexible whereas others are not. Model selection becomes an uphill task. 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 Unifying techniques can result if the reliability growth process is studied with respect the amount of expended testing efforts. In this paper, we propose a generalized framework for deriving several existing as well as new testing effort dependent software reliability growth models with the possibility of imperfect debugging and error generation. The proposed modeling technique is based on the assumption that fault removal/correction is immediate to failure observation and can be extended for the case when there is a clear differentiation between failure observation/detection and fault removal/correction processes. We have also provided a data analysis based on the actual software failure data sets for some of the models discussed and proposed in the paper.

KEYWORDS
Software Reliability Growth Model, Testing Effort function, Imperfect Debugging, Error Generation, Probability Distribution Function

1. INTRODUCTION
With increased complexity of products design, shortened development cycles and highly destructive consequences of software failures, a major responsibility lies in the areas of software debugging, testing and verification. Testing is defined as the execution of a program to find the faults, which might have been introduced in it during various stages of the development cycle. It is also performed to judge the performance, safety, fault-tolerance or security of the software. More importantly, testing provides a mathematical measure of software reliability (i.e., failure/execution time) which forms a vital input to the release decision.
A large number of Software Reliability Growth Models (SRGM), which relate the number of failures (faults identified/corrected) and execution time, have been discussed in the literature [8,11,14]. These SRGM assume diverse testing environment like distinction between failure and correction processes, learning of the testing personnel, possibility of imperfect debugging and fault generation, constant or monotonically increasing / decreasing Fault Detection Rate (FDR) or randomness in the growth curve. 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.

The work in this area started as early as in 1980s with Shantikumar [17] proposing a Generalized birth process model. Gokhale and Trivedi [5] used Testing coverage function to present a unified framework and showed how NHPP based models can be represented by probability distribution functions of fault – detection times. Dohi et al [2] proposed a unification method for NHPP models describing test input and program path searching times stochastically by an infinite server queuing theory. Inoue [6] applied infinite server queuing theory to the basic assumptions of delayed S-shaped SRGM [20] i.e. fault correction phenomenon consists of successive failure observation and detection/correction processes and obtained several NHPP models describing fault correction as a two stage process.

Another unification methodology is based on a systematic study of Fault detection process (FDP) and Fault correction process (FCP) where FCPs are described by detection process with time delay. The idea of modeling FCP as a separate process following the FDP was first used by Schneidewind [16]. More general treatment of this concept is due to Xie et al [18,19] who suggested modeling of Fault detection process as a NHPP based SRGM followed by Fault correction process as a delayed detection process with random time lag. The recent unification scheme (due to Kapur et al [9]) is based on Cumulative Distribution Function for the detection/correction times and incorporates the concept of change point in Fault detection rate. These unification schemes predict the fault content and reliability of the software with respect to the calendar time and do not consider the consumption pattern of...
resources such as computer time, manpower and number of executed test cases etc. More realistic Unifying techniques can result if the reliability growth process is related to the amount of expended testing efforts. In this paper, we propose a generalized framework for deriving several existing as well as new testing effort dependent software reliability growth models with the possibility of imperfect debugging and error generation.

In practical software development scenario, As soon as a failure is observed, the efforts are made to correct the cause of the failure. It is quite possible that the testing team may not be able to remove/correct fault completely 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 removal, the original detected fault is not corrected perfectly. But 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.

It was Goel [3] who first introduced the concept of imperfect debugging. Model due to Obha and Chou [10] is an error generation model applied on G-O model and has been also named as Imperfect debugging model. Kapur and Garg [8] introduced the imperfect debugging in Goel and Okumoto [4]. 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. Pham [13] developed an SRGM for multiple failure types incorporating error generation. Recently, Kapur et al. [7] proposed a flexible SRGM with imperfect debugging and error generation using a logistic function for fault detection rate which reflects the efficiency of the testing/removal team.

In this paper, we present a generalized framework for Software reliability growth modeling with respect to testing effort expenditure and incorporate the concept of 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 proposed models are initially formulated for case when there is no differentiation between failure observation/detection and fault removal/correction processes but can be extended for case when there is a clear differentiation between failure observation/detection and fault removal/correction processes as described in scope for future research.

The existing and new models derived here have been validated and evaluated on two actual software failure data sets. For estimation of parameters of the proposed models, SPSS has been used. The parameter estimation results are quite accurate and encouraging.

Rest of this paper is organized as follows: Section 2 mentions the basic assumptions made followed by the model development under imperfect debugging and error generation. This section also includes many existing and new models which can be derived by following the proposed scheme. Section 3 shows numerical examples for the proposed models based on two real software failure data sets. In section 4 we discuss the scope for future research by considering differentiation between failure observation/detection and fault removal/correction processes. Finally, conclusions are drawn in section 5.

**NOTATIONS**

- \( m(W_t) \) Mean value function (MVF) or the expected number of faults corrected by time \( t \).
- \( a \) Expected number of faults lying dormant in the software when the testing starts i.e at \( t=0 \).
- \( W_t \) Amount of testing effort expended by time \( t \).
- \( a(W_t) \) Total fault content of software dependent on testing effort expended.
- \( \lambda(W_t) \) Intensity function for Fault correction process (FCP) or Fault correction rate per unit time.
- \( G(W_t), F(W_t) \) Testing effort dependent Probability Distribution Function for Failure observation and Fault Correction Times.
- \( g(W_t), f(W_t) \) Testing effort dependent Probability Density Function for Failure observation and Fault Correction Times.
- \( \otimes \) Convolution.
- \( \otimes \) Steiltjes convolution.

**2. PROPOSED MODELLING FRAMEWORK**

**2.1 BASIC ASSUMPTIONS**

The model is based on the following assumptions:

1. Software system is subject to failure during execution caused by faults remaining in the system.
2. The number of faults detected at any time instant is proportional to the remaining number of faults in the software. Each time a failure is observed, immediate correction effort starts and the following may occur:
   (a). Fault content is reduced by one with probability \( p \)  
   (b). Fault content remains unchanged with probability \( 1-p \).
3. During the fault correction process, whether the fault is removed successfully or not, new faults are generated with a constant probability \( \alpha \).
4. The Fault correction times are i.i.d. random variables with probability distribution function

**Proceedings of the 3rd National Conference; INDIACom-2009**
A New Approach For Developing Testing Effort Dependent Software Reliability Growth Models

\[ F(W_t) = \int_0^{W_t} f(x)dx \quad \text{where} \quad F(W_t) \quad \text{is testing effort dependent distributions function.} \]

5. The fault correction process is modeled by NHPP.
6. The initial number of failure observed in the software system at \( t=0 \) is a Poisson random variable with mean of \( a \).

2.2 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

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

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

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

Therefore, we have

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

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

Here it can be noted that

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

From above we obtain

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

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

\[ m(W_t) = E[N(W_t)] = a F(W_t) \]  (4)

By selecting suitable probability distribution function, we can derive MVF for several existing and new Finite failure count models.

From Equation (4), the instantaneous failure intensity function \( \lambda(W_t) \) is given by:

\[ \lambda(W_t) = a F'(W_t) \]

Or we can write

\[
\lambda(W_t) = \frac{dm}{dW_t} = \left[ a - m(W_t) \right] F'(W_t) / [1-F(W_t)] \]

\[ = \left[ a - m(W_t) \right] \lambda(W_t) \]

where \( s(W_t) \) is the failure occurrence rate per remaining fault of the software, or the rate at which the individual faults manifest themselves as failures during testing or hazard rate function. \( [a - m(W_t)] \) denotes the expected number of faults remaining in the software at time \( t \). Since \( [a - m(W_t)] \) is monotonically non-increasing function of time, the nature of the overall failure intensity, \( \lambda(W_t) \), is governed by the nature of failure occurrence rate per fault \( s(W_t) \).

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 time \( t \). That is,

\[ a(W_t) = a + \alpha m(W_t) \]  (6)

Now incorporating the imperfect debugging and error generation in modeling, Equation (5) becomes:

\[
\lambda(W_t) = \frac{dm}{dW_t} = \frac{\left[ a(W_t) - m(W_t) \right] p F'(W_t)}{1-F(W_t)} \]

(7)

where \( p \) is the probability of perfect debugging.

Substituting the value of \( a(W_t) \) from Equation (6) into Equation (7) and solving it using initial condition that at \( t=0, \ m(0) = 0 \ and \ W_0 = 0 \), we get:

\[
m(W_t) = \frac{a}{1-\alpha} \left[ 1 - (1 - F(W_t))^{\rho(1-\alpha)} \right] \]

(8)
Here If \( p=1 \) and \( \alpha = 0 \), i.e. perfect debugging, equation (8) is nothing but equation (4).

### 2.3 MVF For Various New And Existing Models

By substituting different types/forms of distribution functions i.e. different \( F(W_i) \) in Equation (8), we can obtain different mean value functions as given in table 1.

<table>
<thead>
<tr>
<th>Mode 1</th>
<th>Distribution Function ( F(W_i) )</th>
<th>Mean Value Function ( m(W_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRG M-1</td>
<td>( W_i \sim \exp(b) )</td>
<td>( \frac{\alpha}{1-\alpha} \left[ 1 - e^{-b\alpha(1-\alpha)W_i} \right] )</td>
</tr>
<tr>
<td>SRG M-2</td>
<td>( W_i \sim \text{Erlang-2} )</td>
<td>( \frac{\alpha}{1-\alpha} \left[ 1 - (1 + bW_i)e^{-bp(1-\alpha)W_i} \right] )</td>
</tr>
<tr>
<td>SRG M-3</td>
<td>( W_i \sim \text{Erlang-K Stage (b)} )</td>
<td>( \frac{\alpha}{1-\alpha} \left[ 1 - \left( \frac{1}{1+\beta e^{-bW_i}} \right) \right] )</td>
</tr>
<tr>
<td>SRG M-4</td>
<td>( W_i \sim \text{P} ) i.e. ( W_i ) follows a particular kind of distribution whose probability distribution will be given by: ( P(W_i) = \left( 1 + \sum_{k=0}^{\infty} \frac{\beta W_i^k}{k!} \right) e^{-\beta W_i} )</td>
<td>( \frac{\alpha}{1-\alpha} \left[ 1 - \left( \frac{\beta \sum_{k=0}^{\infty} \frac{(bW_i)^k}{k!} e^{-bW_i}}{(1+\beta e^{-bW_i})} \right)^{\rho(1-\alpha)} \right] )</td>
</tr>
<tr>
<td>SRG M-5</td>
<td>( W_i \sim \text{Wei}(b,k) ) (Weibull Distribution)</td>
<td>( \frac{\alpha}{1-\alpha} \left[ 1 - e^{-b\alpha(1-\alpha)W_i^k} \right] )</td>
</tr>
</tbody>
</table>

### 2.4 Modeling Testing Effort

The testing resources spent during debugging 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 [12]:

1. Manpower
2. Computer time.

The key function of manpower engaged in software testing is to 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. 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".

\[
\frac{dW_i}{dt} = \nu(t) [\alpha - W_i]
\]

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

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

\[
W_i = \alpha \left( 1 - e^{-\nu t} \right)
\]

**Case 2:** If \( \nu(t)=\nu t \), we get Rayleigh type curve:

\[
W_i = \alpha \left( 1 - e^{-\nu t^2/2} \right)
\]

**Case 3:** If \( \nu(t)=\nu t \), we get Weibull function:

\[
W_i = \alpha \left( 1 - e^{-\nu t^k} \right)
\]

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

### 3. MODEL VALIDATION, COMPARISON CRITERIA AND DATA ANALYSES

#### 3.1 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 following two real software data sets.

**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 [1].

**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 [12].

#### 3.2 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”?

a. 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:

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

where $k$ is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit [8].

b. Coefficient of Multiple Determination ($R^2$):
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.

$$R^2 = 1 - \frac{SS_{residual}}{SS_{corrected}}.$$  

$R^2$ 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 [8].

c. Bias:
The difference between the observation and prediction of number of failures at any instant of time $i$ is known as PE (prediction error). The average of PEs is known as bias. Lower the value of Bias better is the goodness of fit [15].

d. 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 [15].

e. 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 [15].

3.3 Data Analyses
The SRGM with mean value function $m(t)$ for SRGM-1, 2, 3, 4 and 5 given in Table 1 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. 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 II and Table III. The fitting of the models to DS-1 is graphically illustrated in Fig. 1. SRGM-5 shows a poor fitting to the actual values of the real time data set while all other models fit the data excellently well.

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 and Table V. The fitting of the models to DS-2 is graphically illustrated in Fig. 2. SRGM-3 shows a poor fitting to the actual values of the real time data set while all other models fit the data quite well.

| Table II Parameter Estimates for DS-1 |
|-----------------|--------|------|---|---|--------|
| Models          | a      | b    | k  | p  | β      | α     |
| SRGM-1          | 1193   | .0104| -- | .1362| --     | .2279 |
| SRGM-2          | 1484   | .0956| -- | .0132| --     | .0025 |
| SRGM-3          | 1384   | .0546| -- | .0296| --     | .0214 |
| SRGM-4          | 1513   | .0042| -- | .2813| .1240  | .0000 |
| SRGM-5          | 1446   | .0991| .5718| .1327| --     | .9741 |

| Table III Model Comparison Results |
|-----------------|--------|------|---|---|--------|--------|
| Models          | $R^2$  | MSE  | BIAS | VARIATION | RMSPE   |
| SRGM-1          | .99470 | 1127 | -0.211 | 34.072    | 34.07   |
| SRGM-2          | .99490 | 1085 | -5.193 | 33.013    | 33.42   |
| SRGM-3          | .99335 | 1416 | -12.14 | 36.147    | 38.13   |
| SRGM-4          | .99541 | 977  | -1.672 | 31.683    | 31.73   |
| SRGM-5          | .97470 | 5387 | 17.535129 | 72.312 | 74.41  |

| Table IV : Parameter Estimates for DS-2 |
|-----------------|--------|------|---|---|---|---------|
| Models          | a      | b    | k  | p  | β | α       |
| SRGM-1          | 332    | .1647| -- | .2026| -- | .4110   |
| SRGM-2          | 367    | .4251| -- | .1074| -- | .1247   |
| SRGM-3          | 325    | .4251| -- | .1725| -- | .1201   |
| SRGM-4          | 499    | .0911| -- | .2782| .2324 | .0006   |
| SRGM-5          | 427    | .1779| .8441| .2067| -- | .6478   |
Table V Model Comparison Results for DS-2

<table>
<thead>
<tr>
<th>Models</th>
<th>R²</th>
<th>MSE</th>
<th>BIAS</th>
<th>VARIATION</th>
<th>RMSPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM-1</td>
<td>.98867</td>
<td>116</td>
<td>0.56</td>
<td>11.09</td>
<td>11.11</td>
</tr>
<tr>
<td>SRGM-2</td>
<td>.98704</td>
<td>133</td>
<td>-1.91</td>
<td>11.72</td>
<td>11.87</td>
</tr>
<tr>
<td>SRGM-3</td>
<td>.97679</td>
<td>306</td>
<td>-7.13</td>
<td>16.44</td>
<td>17.92</td>
</tr>
<tr>
<td>SRGM-4</td>
<td>.98970</td>
<td>106</td>
<td>0.224</td>
<td>10.59</td>
<td>10.59</td>
</tr>
<tr>
<td>SRGM-5</td>
<td>.98487</td>
<td>156</td>
<td>1.42</td>
<td>12.76</td>
<td>12.84</td>
</tr>
</tbody>
</table>

Goodness of Fit Curves

Fig. 1: Goodness of fit for DS-1

Fig. 2: Goodness of fit for DS-2

4. SCOPE FOR FUTURE RESEARCH
In this paper we have restricted ourselves to the standard probability distributions e.g. Exponential, Weibull, Erlang-k type. In future the research can be made to work out the use of few other distribution functions like Normal and Gamma for presenting correction times. The mean value function for these distributions will be given by

Also, the methodology proposed for unification of testing effort dependent reliability growth models can be extended to represent the situations when the fault correction is not immediate to the failure observation. There is a time delay between the observation of the fault and the correction of the underlying fault. This time delay can creep in due to various factors e.g. severity / complexity of the faults, change in defect density, skill of the testing team etc. Then FCP is no longer a one-stage process. The correction may be a two / three stage process namely failure observation, fault detection followed by the fault correction. This division of fault correction into different processes defines the complexity of faults present in software. More the delay in removal/correction of a fault on its observation/detection, more complex is the fault. In that case, Equation (5) can be modified as:

$$\lambda(W_t) = \frac{dm}{dW_t} = \frac{(f_g * g)(W_t)}{1 - (F \otimes G)(W_t)} [a - m(W_t)]$$  \hspace{1cm} (9)

or,

$$\lambda(W_t) = h(W_t) [a - m(W_t)]$$

where $h(W_t) = \frac{(f_g * g)(W_t)}{1 - (F \otimes G)(W_t)}$ is the failure observation/detection-fault removal/correction rate.

Now incorporating imperfect debugging and error generation in proposed modeling, we have

$$\lambda(W_t) = \frac{dm}{dW_t}$$  \hspace{1cm} (10)

where

$$\lambda(W_t) = p \frac{(f_g * g)(W_t)}{1 - (F \otimes G)(W_t)} [a + \alpha m(W_t) - m(W_t)]$$

Solving the above differential equation, we get:

$$m(W_t) = \frac{a}{(1 - \alpha)} \left[1 - (F \otimes G)(W_t)\right]^{\frac{1}{\alpha}}$$  \hspace{1cm} (11)

The mean value functions $m(W_t)$ can be derived for different forms of distribution functions $F(W_t)$ and $G(W_t)$.

The numerical ability and accuracy of these models need to be worked out and verified and sets the course of our future research.
5. CONCLUSION
In this paper, a unified framework for testing effort dependent software reliability growth models has been discussed under the assumption that failure observation/detection is immediate to the fault correction process. More realistic software testing scenario has been modeled by incorporating the possibility of two types of imperfect debugging i.e. imperfect debugging and error generation. 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. The scope for future research in this area lies for the case when failure observation/detection, fault removal/correction processes are two/three stage processes. 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. 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.

REFERENCES