A Discrete NHPP Model for Software Reliability Growth with Imperfect Fault Debugging and Fault Generation

P.K. KAPUR*, OM PAL SINGH*, OMAR SHATNAWI#, and ANU GUPTA*

* Department of Operational Research, University of Delhi, Delhi, India
# Department of Computer Science, Prince Hussein bin Abdullah Information Technology College, Mafraq, Jordon

(Received on February 10, 2006)

Abstract: This paper presents a discrete software reliability growth model (SRGM) and introduces the concept of two types of imperfect debugging during software fault removal phenomenon with Logistic Fault removal rate. Most of the discrete SRGMs discussed in the literature seldom differentiate between the failure observation and fault removal processes. In real software development environment, the number of failures observed need not be same as the number of error removed. If the number of failures observed is more than the number of faults removed then we have the case of imperfect debugging. Due to the complexity of the software system and the incomplete understanding of the software requirements, specifications and structure, the testing team may not be able to remove the fault perfectly on the detection of the failure and the original fault may remain or get replaced by another fault. While the first phenomenon is known as imperfect fault debugging, the second is called fault generation. In case of imperfect fault debugging the fault content of the software is not changed, but just because of incomplete understanding of the software, the detected fault is not removed completely. But in case of error generation the fault content increases as the testing progresses and removal results in introduction of new faults while removing old ones. The model has been validated, evaluated and compared with other existing discrete NHPP models by applying it on actual failure / fault removal data sets cited from real software development projects. The results show that the proposed model provides improved goodness of fit and predictive validity for software failure / fault removal data.

Key Words: software reliability, software reliability growth model (SRGM), non-homogeneous Poisson process (NHPP), software testing, test occasions (cases).

Acronyms**
SRGM Software Reliability Growth Model
NHPP Non-homogeneous Poisson Process
Computer systems now pervade every aspect of our daily lives. While this has benefited society and increased our productivity, it has also made our lives more critically dependent on their correct functioning. Software reliability assessment is important to evaluate and predict the reliability and performance of software systems. Several SRGMs have been developed in the literature to estimate the fault content and fault removal rate per fault in software. Goel and Okumoto [3] have proposed NHPP based SRGM assuming that the failure intensity is proportional to the number of faults remaining in the software. The model is very simple and can describe exponential failure curves. Ohba [14] refined the Goel-Okumoto model by assuming that the fault detection/removal rate increases with time and that there are two types of faults in the software. SRGM proposed by Bittanti et al. [1] and Kapur and Garg [8] have similar forms as that of Ohba [14] but are developed under different sets of assumptions. Bittanti et al. [1] proposed an SRGM exploiting the fault removal (exposure) rate during the initial and final time epochs of testing. Kapur and Garg [8] describe a fault removal phenomenon, where they assume that during a fault removal process, some of the remaining faults may also be removed. These models can describe both exponential and S-shaped growth curves and therefore are termed as flexible models.

NHPP based SRGMs are generally classified into two groups. The first group contains models, which use the execution time (i.e., CPU time) or calendar time. Such models are called continuous time models. The second group contains models, which use the test cases as a unit of fault removal period. Such models are called discrete time models, since the unit of software fault removal period is countable (Inoue and Yamada [5]; Kapur et al. [9]; Pham [15]; Musa [13]). A test case can be a single computer test run executed in an hour, day, week or even month. Therefore, it includes the computer test run and length of time spent to visually inspect the software source code. A large number of models have been developed in the first group while fewer are there in the second group due to the difficulties in terms of mathematical complexity involved.

Lately attempts have been made to develop flexible discrete SRGMs. In this paper a discrete flexible SRGM is developed using Probability Generating Function (P.G.F) incorporating two types of imperfect debugging i.e. Error generation and imperfect fault debugging. It is further shown, how continuous time SRGM can be derived from the discrete model.
The utility of discrete reliability growth models cannot be underestimated. As the software failure data sets are discrete, these models many times provide better fit than their continuous time counterparts. Therefore, in spite of difficulties in terms of mathematical complexity involved, discrete models are proposed regularly. Most of discrete models discussed in the literature seldom differentiate between the failure observation and fault removal processes. In real software development scene, the number of failure observed can be less than or more than the number of error removed. Kapur and Garg [8] has discussed the first case in their Error removal phenomenon flexible model which shows as the testing grows and testing team gain experience, additional number of faults are removed without them causing any failure. But if the number of failure observed is more than the number of error removed then we are having the case of imperfect debugging.

Due to the complexity of the software system and the incomplete understanding of the software requirements, specifications and structure, the testing team may not be able to remove the fault perfectly on the detection of the failure and the original fault may remain or replaced by another fault. While the first phenomenon is known as imperfect debugging, the second is called fault generation. In case of imperfect debugging the fault content of the software is not changed, but because of incomplete understanding of the software, the detected fault is not removed completely. But in case of error generation the fault content increases as the testing progresses and removal results in introduction of new faults while removing old ones.

The concept of imperfect debugging was first introduced by Goel [4]. He introduced the probability of imperfect debugging in Jelinski and Moranda [6]. Kapur and Garg [10] introduced the imperfect debugging in Goel and Okumoto [3]. They assumed that the FRR per remaining faults is reduced due to imperfect debugging. Thus the number of failures observed by time infinity is more than the initial fault content. Although these two models describe the imperfect debugging phenomenon yet the software reliability growth curve of these models is always exponential. Moreover, they assume that the probability of imperfect debugging is independent of the testing time. Thus, they ignore the role of the learning process during the testing phase by not accounting for the experience gained with the progress of software testing. Actually, the probability of imperfect debugging is supposed to be a maximum in the early stage of the testing phase and is supposed to reduce with the progress of testing. All these models are continuous time models. Kapur et al. [11] have proposed three discrete models taking into account imperfect fault debugging and fault generation phenomena separately.

In this paper, a general discrete SRGM incorporating fault generation and imperfect debugging with learning has been proposed. The proposed model has been validated and evaluated on actual software failure / fault removal DS and compared with other discrete models. The importance and utility of discrete time modeling have been highlighted.

2. Discrete SRGMs Based On NHPP - A General Description
During the software testing phase a software system is executed with a sample of test cases to detect and correct software faults, which cause failures. A discrete counting process \( [N(n), n \geq 0], (n = 0, 1, 2, \ldots) \) is said to be an NHPP with mean value function \( m(n) \), if it satisfies the following conditions:

There are no failures experienced at \( n=0 \), that is, \( N(0) = 0 \).

The counting process has independent increments, that is, the number of failures experienced during \( (n,n+1)^{th} \) test cases is independent of the history and implies that \( m(n+1) \) of the process depends only on the present state \( m(n) \) and is independent of its past state \( m(x) \), for \( x < n \).

In other words, for any collection of the numbers of test cases \( n_1, n_2, \ldots, n_k (0 < n_1 < n_2 < \ldots < n_k) \), the \( k \) random variables \( (N(n_1), N(n_2) - N(n_1), \ldots, N(n_k) - N(n_{k-1})) \) are statistically independent.

For any of two numbers of test cases \( n_i \) and \( n_j \) where \( 0 \leq n_i \leq n_j \), we have:

\[
\Pr \{ N(n_j) - N(n_i) = x \} = \frac{\{m(n_j) - m(n_i)\}^x}{x!} \exp\left[ - \{m(n_j) - m(n_i)\} \right], \quad x = 0, 1, 2 \tag{1}
\]

The mean value function \( m(n) \) which is a non-decreasing in \( n \) represents the expected cumulative number of faults detected by \( n \) test cases. Then the NHPP model with \( m(n) \) is formulated by:

\[
\Pr \{ N(n) = x \} = \frac{\{m(n)\}^x}{x!} \exp\left[ - \{m(n)\} \right] \tag{2}
\]

Let \( \overline{N}(n) \) denote the number of faults remaining in the system after execution of the \( n^{th} \) test run. Then we have:

\[
\overline{N}(n) = N(\infty) - N(n) \tag{3}
\]

where \( N(\infty) \) represents the total initial fault content of the software. The expected value of \( \overline{N}(n) \) is given by:

\[
E(n) = m(\infty) - m(n) \tag{4}
\]

where \( m(\infty) \) represents the expected number of faults to be eventually detected. Suppose that \( n_d \) faults have been detected by \( n \) test cases. The conditional distribution of \( \overline{N}(n) \), given that \( \overline{N}(n) = n_d \), is given by:
A Discrete NHPP Model for Software Reliability Growth

\[
\Pr\{N(n) = y | N(n) = n_d\} = \frac{\{E(n)\}^y}{y!} \exp\{-\{E(n)\}\}
\]  

(5)

Now, the probability of no faults detected between the \(n^{th}\) and \((n+h)^{th}\) test cases, given that \(n_d\) faults have been detected by \(n\) test cases, is given by:

\[
R(h/n) = \exp\{-m(n+h) - m(n)\}, \quad n, h = 0,1,2,\ldots
\]

(6)

The above function \(R(h/n)\) is called a software reliability function based upon an NHPP for a discrete SRGM and is independent of \(n_d\).

In the following section a new SRGM is proposed.

3. Software Reliability Modeling

3.1. Development

During debugging process faults are identified and removed upon failures. In reality this may not be always true. The corrections may themselves introduce new faults or they may inadvertently create conditions, not previously experienced, that enable other faults to cause failures. This results in situations where the actual fault removals are less than the removal attempts. Therefore, the FRR is reduced by the probability of imperfect debugging. Besides, there is a good chance that some new faults might be introduced during the course of debugging (Yamada et al. [18], Kapur et al. [9, 11], and Pham [15]). Learning occurs if testing appears to improve dynamically in efficiency as one progresses through the testing phase. The model incorporates the logistic learning function during the fault removal phase to represent the learning that grows as the testing progresses. By assuming the FRR is dependent on the number of test cases, the role of the learning process during the testing phase can be established. The model proposed below incorporates these three factors.

3.2. Assumptions

The model developed below is based upon the following basic assumptions:

1. Failure observation / fault removal phenomenon is modeled by NHPP.
2. Software is subject to failures during execution caused by faults remaining in the software.
3. Each time a failure is observed, an immediate effort takes place to decide the cause of the failure in order to remove it.
4. Failure rate is equally affected by faults remaining in the software.
5. The debugging process is imperfect.
6. FRR is a logistic learning function dependent on the number of test cases.

3.3. Notation
\[ m_r(n) = m_r(n-1) + b_r P_{n-1} \]

where \( m_r(n) \) is the number of faults removed after the execution of the \( n \)th test run and both \( a(n) \) and \( b(n+1) \) are dependent on the number test cases. An increasing \( a(n) \) implies an increasing total number of faults, and thus reflects fault generation. Whereas, \( b(n+1) \) is a logistic learning function and is affected by the probability of fault removal on a failure.

Let us define:

\[ a(n) = a_0 (1 + \alpha \delta)^n \]

\[ b(n+1) = \frac{b_0 p}{1 + \beta (1 - b_0 p \delta)^{n+1}} \]

For derivation of \( b(n) \) as a logistic function refer the appendix A.

By substituting the above forms of \( a(n) \) and \( b(n+1) \) in the difference equation (7) we get:

\[ \frac{m_r(n) - m_r(n-1)}{\delta} = \frac{b_r P_{n-1}}{1 + \beta (1 - b_0 p \delta)^{n+1}} (a_0 (1 + \alpha \delta)^n - m_r(n)) \]

To solve this difference equation we use the technique of Probability Generating Function (P.G.F.). For detailed solution for deriving mean value function \( m_r(n) \), refer the appendix B.

The closed form solution is as given below:

\[ m_r(n) = \frac{a_0 b_0 p \delta}{1 + \beta (1 - b_0 p \delta)^n} \left[ \frac{(1 + \alpha \delta)^n - (1 - b_0 p \delta)^n}{(\alpha \delta + b_0 p \delta)} \right] \]
A Discrete NHPP Model for Software Reliability Growth

where, \( m_r(n=0)=0 \) and \( m_r(n=\infty)=\infty \).

Here, it can be observed that if fault generation rate \( \alpha = 0 \) i.e. if there is no new faults are introduced during the debugging process, then \( m_r(n) \) given by expression (10) reduces to:

\[
m_r(n) = a_0 \left[ \frac{1 - (1 - b_0 p \delta)^n}{1 + \beta (1 - b_0 p \delta)^n} \right]
\]

It can be further observed that, in this case, the total number of faults which can be removed eventually is same as the initial fault content \( a_0 \) i.e. If \( \alpha = 0 \), then \( m_r(n=\infty)=a_0 \).

3.5. Derivation of Equivalent Continuous Model

To derive the equivalent continuous SRGM corresponding to discrete mean value function given by (10), we take limit \( \delta \to 0 \).

Let us define \( t = n\delta \).

We know that \( \lim_{x \to 0} \frac{1}{(1 + x)^x} = e \)

As \( \delta \to 0 \)

\[
a(n) = a_0(1 + \alpha \delta)^n \to a_0 e^{\alpha t} = a(t)
\]

\[
b(n+1) = \frac{b_0 p}{1 + \beta (1 - b_0 p \delta)^n} \to \frac{b_0 p}{1 + \beta e^{-b_0 p t}} = b(t)
\]

The corresponding differential equation is given by:

\[
\frac{d}{dt} m_r(t) = b(t) \left( a(t) - m_r(t) \right)
\]

As \( \delta \to 0 \)

\[
\frac{a_0 b_0 p \delta}{1 + \beta (1 - b_0 p \delta)^n} \left[ (1 + \alpha \delta)^n - (1 - b_0 p \delta)^n \right] \to \frac{a_0 b_0 p}{1 + \beta e^{-b_0 p t}} \left[ e^{\alpha t} - e^{-b_0 p t} \right]
\]

which is same as the solution of above differential equation with initial condition \( m_r(t=0)=0 \) and this is an extension of Kapur and Garg model [8] with imperfect fault removal and fault generation. If we set \( \alpha = 0 \) and \( p = 1 \), i.e., no fault generation and perfect debugging, then the proposed model is nothing but Kapur and Garg model [8].

This proposed model is very interesting from various points of view. Besides its interpretation as a flexible S-shaped fault removal model, this model has the exponential
model (Yamada and Osaki [17]) and the imperfect debugging model (Kapur et al. [11]) as special cases. Note that this proposed model is able to model both cases of strictly decreasing failure intensity and the case of increasing-and-decreasing failure intensity. None of the exponential model (Yamada and Osaki [17]) and the ordinary delayed S-shaped model (Kapur et al. [9]) can do both.

4. Estimation of Parameters

Parameters estimation is of primary concern in software reliability prediction. For this, the failure data is collected and is recorded in either of the following two formats—the first approach is to record the time between successive failures while second way is to collect the number of failures experienced at regular testing intervals. If failure data is available then the values of the unknown parameters can be estimated by using either maximum Likelihood Method or by using the technique of least square method. The brief description of these two techniques is:

Maximum Likelihood Method: The MLE procedure when the failure data is given in the form $(n_i, x_i), i=1,2,3...k$, where $x_i$ is the cumulative number of faults removed by $n_i$ test cases ($0 < n_1 < n_2 < ... < n_k$) and $n_i$ is the accumulated test runs spent to remove $x_i$ faults. The Likelihood function $L$ is given as:

$$L(Parameters| (n_i, x_i)) = \prod_{i=1}^{k} \frac{[m(n_i) - m(n_{i-1})]^{x_i-x_{i-1}}}{(x_i-x_{i-1})!} e^{-m(n_i)}$$

The MLE of the parameters of SRGMs can be obtained by maximizing $L$ with respect to the model parameters.

Least square method: In this method, the sum of square of the difference between observed value and the value estimated by the model is minimized. If the failure data consists of $k$ pairs of sample values $(n_i, x_i), i=1,2,3...k$, where $x_i$ is the cumulative number of faults removed by $n_i$ test cases ($0 < n_1 < n_2 < ...{n_k}$) and $n_i$ is the accumulated test runs spent to remove $x_i$ faults. Let the estimated value of the number of faults removed by $n_i$ test cases be $\hat{m}(n_i)$. Then parameter estimation by least square method consists of minimizing the sum of squares of the deviation between actual and estimated values i.e.

$$\text{Sum of Squares} = \sum_{i=1}^{k} (\hat{m}(n_i) - x_i)^2$$

Bayesian Analysis: When no failure data or very small amount of the failure data is available then it is not possible to estimate the values of the parameters by using above two specified techniques. In such case, the parameters are not assumed to be fixed at some unknown value, but they are assumed to follow some probability distribution, known as prior distribution. Given the software reliability model and the assumption about the distribution of the model parameters, it is possible to obtain the distribution of random variable $N(n)$ (known as posterior distribution) and its expected value $m(n)$ i.e. mean value function.
4.1 Parameter Estimation for the proposed model

In this paper, the maximum likelihood method is used to estimate the parameters \((a_0, b_0, p, \alpha, \beta)\) of the proposed model. Since the DS used in this paper are given in the form of pairs \((n_i, x_i), i=1,2,3...k\), where \(x_i\) is the cumulative number of faults removed by \(n_i\) test cases \((0<n_1<n_2<...<n_k)\) and \(n_i\) is the accumulated test runs spent to remove \(x_i\) faults. The Likelihood function \(L\) is given as:

\[
L(Parameters|n_i, x_i) = \prod_{i=1}^{k} \frac{[m(n_i) - m(n_{i-1})]^{x_i - x_{i-1}}}{(x_i - x_{i-1})!} e^{-m(n_i)}
\]

(11)

The likelihood function or the log Likelihood function of (11) can be maximized with respect to the parameters to find their estimates. Following constraints can also be used: \(a_0>0\), \(0<b_0<1\), \(0<p\leq 1\), \(\alpha\geq 0\), \(\beta\geq 0\).

Taking natural logarithm of (11) we get:

\[
\ln L = \sum_{i=1}^{k} (x_i - x_{i-1}) \ln \left[\frac{m(n_i) - m(n_{i-1})}{m(n_k) - m(n_{k-1})}\right] - \sum_{i=1}^{k} \ln (x_i - x_{i-1})
\]

(12)

The MLE of the parameters of SRGMs can be obtained by maximizing (11) or (12) with respect to the model parameters.

5. Model Validation

To check the validity of the proposed model to describe the software reliability growth, it has been tested on three DS. The DS-I is cited from (Brooks and Motley [2]) in which 2362 faults were detected after testing for 15 weeks. The DS-II is for a radar system of size 124 KLOC (Brooks and Motley [2]) in which 1301 faults were detected after testing for 35 months. The DS-III was collected from test of system T1 (Musa et al. [13]) in which 136 faults were detected after testing for 21 weeks.

5.1. Model Evaluation

The performance of SRGM is judged by their ability to fit the past software failure occurrence / fault removal data and to predict satisfactorily the future behavior of the software failure occurrence / fault removal process (Musa et al. [13], Kapur et al. [9]). Therefore, we use two types of comparison criteria:

1. The Goodness of Fit Criteria.
2. The Predictive Validity Criterion.

5.2. The Goodness of Fit Criteria

The Sum of Squared Error (SSE): SSE measures the distance of a model estimate value from the actual data, as follows:
Where $k$ is the number of observations, $\hat{m}(n_i)$ is the estimated cumulative number of failures by $n_i$ test case obtained from the fitted mean value function (i.e., SRGM), and $X_i$ is the total number of failures observed by $n_i$ test cases. Lower value of $SSE$ indicates less fitting error, thus better goodness of fit.

The Akaike Information Criterion (AIC): It is defined as:

$$AIC = -2 \times \log(\text{max. of likelihood function}) + 2 \times N$$

Where $N$ is the number of the parameters used in the model. Lower value of AIC indicates more confidence in the model thus a better-fit and predictive validity.

In other words, we evaluate the performance of the models using $SSE$ and AIC metrics. The smaller the metric value the better the model fits relative to other models run on the same DS.

R Squared ($R^2$): Goodness-of-fit measure of a linear model, sometimes called the coefficient of determination. It is the proportion of variation in the dependent variable explained by the regression model. It ranges in value from 0 to 1. Small values indicate that the model does not fit the data well.

5.3. Predictive Validity Criterion

Predictive validity is defined as the ability of the model to determine the future failure behavior from present and past failure behavior. This criterion was proposed by Musa et al. [13]. Suppose $n_k$ be the last test case, $x_k$ is number of faults detected during the interval $(0, n_k]$, and $\hat{m}(n_k)$ is the estimated value of the mean value function $m_r(n)$ at $n_k$, which is determined using the actually observed data up to an arbitrary test case $n_k (0 < n_k \leq n_k^*)$, in which $n_k^*$ denotes the testing progress ratio. In other words, the number of failures by $n_k$ can be predicted by the SRGM and then compared with the actually observed number $x_k$. The difference between the predicted value $\hat{m}(n_k)$ and the reported value $x_k$ measures the prediction fault. The ratio $\left(\frac{\hat{m}(n_k) - x_k}{x_k}\right)$ is called Relative Prediction Error (RPE). If the RPE value is negative / positive the SRGM is said to underestimate / overestimate the future failure phenomenon. A value close to zero for RPE indicates more accurate prediction, thus more confidence in the model and better predictive validity. The value of RPE is said to be acceptable if it is within $\pm(10\%)$ (Kapur et al. [9]).

6. Data Analysis and Model Comparison
6.1. Goodness of Fit Analysis

Using MLE method, the estimation values of the model parameters for all DS are given in table I. The fitting of the model to all DS is graphically illustrated in figures 1, 2 & 3. It is clearly seen from the figures that the model fits all DS excellently. Comparison of the proposed model and other well-documented discrete SRGM due to Yamada et al. [17] and Kapure et al. [9] based on NHPP in terms of goodness of fit for all DS has been worked out. The results are presented in tables II, III & IV. It is clearly seen from the tables that the proposed model is the best among the models under comparison in terms of SSE, AIC & $R^2$.

6.2. Predictive Validity Analysis

All the DS are truncated into different proportions and used to estimate the parameters of the proposed model. For each truncation, one value of RPE is obtained and given in tables V, VI & VII. The tables give the results of the predictive validity. It is observed that the predictive validity of the model varies from one truncation to another. It is clearly seen from both the tables V & VII that 50% of the total test time is sufficient to predict the future reasonably and from the table VI that 65% of the total test time is sufficient to predict the future reasonably.

Figures 4, 5 & 6 illustrate graphically the retrodictive and predictive ability of the proposed model. In each case the DS-I & III are truncated at $n_e$ (50% approx.) and the DS-II is truncated at $n_e$ (65% approx.) to estimate the model parameters. The model is then used to estimate the whole DS. The points below $n_e$ (marked by the intersection of the horizontal line with the curve) demonstrate the retrodictive ability while the points above $n_e$ demonstrate the predictive ability of the model.

6.3. Estimation Results

<table>
<thead>
<tr>
<th>Model</th>
<th>DS</th>
<th>Parameters Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$a_0$</td>
</tr>
<tr>
<td>Proposed</td>
<td>DS-I 2629</td>
<td>0.1018</td>
</tr>
<tr>
<td></td>
<td>DS-II 1331</td>
<td>0.1859</td>
</tr>
<tr>
<td></td>
<td>DS-III 151</td>
<td>0.9969</td>
</tr>
</tbody>
</table>

Consider the estimated values of the parameters given in Table I. Here $a_0$ represents the initial fault content of the software; $b_0$ is limiting value of FDR; $p$ is the probability of perfect debugging; $\alpha$ is error generation rate and $\beta$ is a constant to represent the learning phenomenon in FDR. If we carefully scan the estimation results for DS-I, we observe that $p = 1$. It represents the case of perfect debugging i.e. every detected fault is removed with certainty. Here $\alpha = 0.00901$ which is very low. The parameter $\beta = 0$ signifies the absence of any learning
gained by the testing team. For DS-II, \( p = 0.9772 \). It represents approx. 98% of the detected faults are removed perfectly while only a little more than 2% faults remain even after being detected. Here \( \alpha = 0 \) i.e. no fault generation. The learning is high as represented by \( \beta = 20.13 \). For DS-III \( p = 0.9969 \) which represents 99.69% of the detected faults are removed perfectly while only a very small fraction 0.31% faults remain even after being detected. Here \( \alpha = 0 \) i.e. no fault generation. The learning is high with \( \beta = 585 \). The large value of \( \beta \) also justifies the highly S-shaped nature of the dataset DS-III.

Table II. (Goodness of Fit for DS-I)

<table>
<thead>
<tr>
<th>Models under Comparison</th>
<th>Parameter Estimation</th>
<th>Comparison Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( a_0 )</td>
<td>( b_0 )</td>
</tr>
<tr>
<td>Yamada et al. [17]</td>
<td>3004</td>
<td>0.0897</td>
</tr>
<tr>
<td>Kapure et al. [9]</td>
<td>2287</td>
<td>0.2607</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>See Table I.</td>
<td></td>
</tr>
</tbody>
</table>

Table III. (Goodness of Fit for DS-II)

<table>
<thead>
<tr>
<th>Models under Comparison</th>
<th>Parameter Estimation</th>
<th>Comparison Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( a_0 )</td>
<td>( b_0 )</td>
</tr>
<tr>
<td>Yamada et al. [17]</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Kapure et al. [9]</td>
<td>1735</td>
<td>0.0814</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>See Table I.</td>
<td></td>
</tr>
</tbody>
</table>

Table IV. (Goodness of Fit for DS-III)

<table>
<thead>
<tr>
<th>Models under Comparison</th>
<th>Parameter Estimation</th>
<th>Comparison Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( a_0 )</td>
<td>( b_0 )</td>
</tr>
<tr>
<td>Yamada et al. [17]</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Kapure et al. [9]</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>See Table I.</td>
<td></td>
</tr>
</tbody>
</table>

* Indicates that the model fails to give any plausible result.

It can be observed from table III, that for DS-II, while the proposed model has shown remarkable improvement over model due to Kapure et al. [9] whereas, Yamada et al. [17] model fails to give any plausible results. For DS-III, both the models under comparison have failed to yield any feasible values of the parameters, but the proposed model yield the results with high value of \( R^2 \) and low values of SSE, AIC.

Table V, VI, VII provide predictive validity of the proposed discrete model. For DS-I, we observe that even 50% of data is sufficient to predict the future failure behaviour well with RPE as low as 4.15%. It is clearly seen from the table VI that for DS-II, 65% of the total test
A Discrete NHPP Model for Software Reliability Growth

data helps to predict the future well with relative prediction error as low as 12.41%. Similar to DS-I, for DS-III only 50% data provides good estimates of the future with RPE -05.40%.

<table>
<thead>
<tr>
<th>Model</th>
<th>( \left( n_i/n_k \right) )</th>
<th>( \hat{m}(n_k) )</th>
<th>( RPE ) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Model</td>
<td>100% 2279.9</td>
<td>-3.47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>95% 2446.9</td>
<td>3.60</td>
<td></td>
</tr>
<tr>
<td></td>
<td>85% 2380.2</td>
<td>0.77</td>
<td></td>
</tr>
<tr>
<td></td>
<td>75% 2341.9</td>
<td>-0.85</td>
<td></td>
</tr>
<tr>
<td></td>
<td>65% 2055.9</td>
<td>-12.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50% 2460.1</td>
<td>4.15</td>
<td></td>
</tr>
</tbody>
</table>

| Table V. (Predictive Validity for DS-I) |

<table>
<thead>
<tr>
<th>Model</th>
<th>( \left( n_i/n_k \right) )</th>
<th>( \hat{m}(n_k) )</th>
<th>( RPE ) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Model</td>
<td>100% 1306.4</td>
<td>00.42</td>
<td></td>
</tr>
<tr>
<td></td>
<td>95% 1310.2</td>
<td>00.71</td>
<td></td>
</tr>
<tr>
<td></td>
<td>85% 1316.1</td>
<td>01.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>75% 1346.4</td>
<td>03.49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>65% 1462.4</td>
<td>12.41</td>
<td></td>
</tr>
</tbody>
</table>

| Table VI. (Predictive Validity for DS-II) |

<table>
<thead>
<tr>
<th>Model</th>
<th>( \left( n_i/n_k \right) )</th>
<th>( \hat{m}(n_k) )</th>
<th>( RPE ) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Model</td>
<td>100% 138.2</td>
<td>01.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td>95% 140.4</td>
<td>03.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>85% 144.8</td>
<td>06.46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>75% 155.5</td>
<td>14.37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>65% 160.9</td>
<td>18.27</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50% 128.1</td>
<td>-05.40</td>
<td></td>
</tr>
</tbody>
</table>

| Table VII. (Predictive Validity for DS-III) |

6.4. Goodness of Fit Curves for all Datasets

The goodness of fit for the proposed model corresponding to datasets DS-I, II, III is graphically presented in Figures 1, 2 and 3 respectively. The graphs have been plotted between actual and estimated values for the cumulative number of faults for the three data sets under consideration. The curves show excellent fit for the proposed model with the estimated values very near to observed failure data.

Figures 4, 5 and 6 depict graphically the retrodictive and predictive ability of the proposed model. For DS-I & III, data is truncated to approx. 50% level and for DS-II, the truncation is
done till 65\% to estimate the model parameters. The points below truncation point show the retrodictive ability while the points above it indicate the predictive ability of the model.

From these three figures, it can be observed that the proposed discrete model not only fits the past well but also predicts the future reasonably well.
7. Conclusion

In this paper, a new developed discrete SRGM with two types of imperfect debugging has been presented. The first type, known as error generation, describes the situation when each error removal attempt increases the fault content of the software. The second type, less damaging, is the case of imperfect debugging where all detected errors are not removed completely. Here the numbers of removal attempts are more than actual fault content but imperfect debugging does not change the content of faults in the software. The concept of learning has been incorporated in the FRR to show the gain in experience and improvement in the testing efficiency of the team as the testing grows. The model has been validated, evaluated, and compared with other existing discrete NHPP models by applying it on three DS. The results show that the proposed model provides improved goodness of fit and predictive validity for software failure occurrence / fault removal data due to its applicability and flexibility.

Acknowledgement
Thanks are due to anonymous referees who helped improve the paper and its presentation.

References


Appendix A

Consider the following difference equation:

\[
\frac{b(n+1)-b(n)}{\delta} = b(n+1) \left( b_0 p - b(n) \right)
\]

To solve this difference equation we use the technique of Probability Generating Function.

Multiply both sides with \( z^n \) and taking summation over \( n = 0 \) to \( n = \infty \) we get:

\[
\sum_{n=0}^{\infty} z^n b(n+1) - \sum_{n=0}^{\infty} z^n b(n) = b_0 p \delta \sum_{n=0}^{\infty} z^n b(n+1) - \delta \sum_{n=0}^{\infty} z^n b(n) b(n+1)
\]

Denote \( \sum_{n=0}^{\infty} z^n b(n) = G(z) \) and solving we get:

\[
\left[ (1-b_0 p \delta)-z \right] G(z) = -\delta \sum_{n=0}^{\infty} z^{n+1} b(n) b(n+1)
\]

\[
\left[ (1-b_0 p \delta)-z \right] \left[ b(0)+z b(1)+z^2 b(2) + \ldots \right]
\]

\[
= -\delta \left[ z b(0) b(1) + z^2 b(1) b(2) + \ldots \right]
\]

Comparing the coefficients of like powers of \( z \) on both sides, the closed form solution is given by:

\[
b(n) = \frac{b_0 p}{1+\beta (1-b_0 p \delta)}^n
\]

where, \( \beta = \frac{k-b(0)}{b(0)} \) and \( b(0) \neq 0 \)

Appendix B

Consider the following difference equation:

\[
\frac{m_j(n+1)-m_j(n)}{\delta} = \frac{b_0 p}{1+\beta (1-b_0 p \delta)^{n+1}} \left( a_0 (1+\alpha \delta)^n - m_j(n) \right)
\]

To solve this, multiply both sides with \( z^n \) and taking summation over \( n = 0 \) to \( n = \infty \) we get:
\[
\sum_{n=0}^{\infty} z^n m_r(n+1) - \sum_{n=0}^{\infty} z^n m_r(n) = \sum_{n=0}^{\infty} z^n \frac{b_0 p \delta}{1 + \beta(1 - b_0 p \delta)^n} \left( a_0 (1 + \alpha \delta)^n - m_r(n) \right)
\]

Denote \( \sum_{n=0}^{\infty} z^n m_r(n) = P(z) \) and solving we get:

\[
(1-z)P(z) = \sum_{n=0}^{\infty} z^{n+1} \frac{b_0 p \delta}{1 + \beta(1 - b_0 p \delta)^n} \left( a_0 (1 + \alpha \delta)^n - m_r(n) \right)
\]

\[
(1-z) \left[ z m_r(0) + z^2 m_r(2) + z^3 m_r(3) + \ldots \right] = \frac{b_0 p \delta z}{1 + \beta \left( 1 - b_0 p \delta \right)^n} \left( a_0 (1 + \alpha \delta)^n - m_r(n) \right) + \ldots
\]

Comparing the coefficients of like powers of \( z \) on both sides with initial condition \( m_r(0) = 0 \) the closed form solution is as given below:

\[
m_r(n) = \frac{a_0 b_0 p \delta}{1 + \beta \left( 1 - b_0 p \delta \right)^n} \left[ \frac{(1 + \alpha \delta)^n - (1 - b_0 p \delta)^n}{(\alpha \delta + b_0 p \delta)} \right]
\]

P. K. Kapur is Professor and former Head of the Department of Operational Research, University of Delhi. He obtained his Ph.D. degree from University of Delhi in 1977. He has published extensively in Indian journals and abroad in the areas of hardware reliability, maintenance, optimization, queueing theory and software reliability. His papers have appeared in \textit{IEEE Transactions on Reliability}, \textit{Asia Pacific Journal of Operational Research}, \textit{R.A.I.R.O.}, \textit{Microelectronics and Reliability}, \textit{Journal of Software Engineering}, \textit{International Journal of System Science}, and many reputed international journals. He has edited three volumes on operations research and co-authored a book "\textit{Contributions to Hardware and Software Reliability}" published from World Scientific, Singapore. He has edited a special issue of \textit{International Journal of Performability Engineering}, \textit{International Journal of Quality Reliability and Safety Engineering} and \textit{OPSEARCH}. He has organized two International Conferences on Quality Reliability and Information Technology. He has guided M. Tech / Ph.D. theses in Computer Science as well as in Operations Research. He is currently the President of Operational Research Society of India.