Mathematical Modelling In Software Reliability

OMAR SHATNAWI1 MOHAMMAD TAIB SHATNAWI2
1Department of Computer Science, Al al-Bayt University, Mafraq 25113, JORDAN
2Al-Huson University College, Al-Balqa’ Applied University, Irbid 21110, Jordan
{dromali@lycos.com}

Abstract: A mathematical model based on stochastic and statistics theories is useful to describe the software fault-removal phenomena or the software failure-occurrence phenomena and estimate the software reliability quantitatively. A mathematical tool which describes software reliability aspect is a software reliability growth model (SRGM). Discrete time models in software reliability are important and a little effort has been made in this direction. Their importance cannot be underestimated since the number of test cases is more appropriate measure of the fault removal/detection period than the CPU/calendar time used by continuous time model. These models generally provide a better fit than their continuous time counterparts. It is important to note that due to the complexity of software design, it is not expected that any single model can incorporate all factors which are thought to influence software reliability. In this paper, we show how beginning with very simple assumptions, non-homogenous Poisson process (NHPP) type of discrete time SRGMs, are gradually made more realistic with the incorporation of imperfect debugging, involvement of a learning-process in debugging and introduction of new faults. The applicability of the resultant generalized model is demonstrated through several actual software reliability data sets obtained from different software development projects. The proposed generalized model is also checked against different components of the model, including existing one, thus highlighting its applicability.

Keywords: Software engineering, software testing, software reliability, imperfect debugging, fault generation, test cases, probability generation function.

1 Introduction
Software reliability engineering is rapidly emerging as an important field of study in the area of information technology. Mathematical models play a significant role in its growth. These models provide quantitative tools to assess the reliability of the developers’ software. Numerous software reliability growth models (SRGMs), which report the number of failures (faults identified/removed), have been discussed in the literature [1-4]. An SRGM provides a mathematical relationship between time span of testing or using the software and the cumulative number of faults detected. It is used to assess the reliability of the software during testing and operational phases. These models are used to predict the fault-content and reliability of the software. Generally the SRGMs are classified into two groups. The first group contains models, which use the machine execution CPU/calendar time as a unit of fault detection period. Such models are called continuous time models. The second group contains models, which use the number of test occasions/cases as a unit of fault detection period. Such models are called discrete time models, since the unit of software fault detection period is countable. 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 inspect the software source code visually. A large number of models have been developed in the first group, while fewer are in the second group owing to the difficulties caused by their mathematical complexity [3,5,6].

In most discrete time models the fault removal process is assumed to be perfect. But 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 (imperfect debugging) and the original fault may remain or replaced by another fault (fault generation). The learning-process of software developers is closely related to the changes in the efficiency of testing during a testing phase and usually manifests itself as a changing fault detection rate [6,7]. In the software reliability literature, most of these issues are addressed using continuous time modelling approach [8-13]. This paper addresses all of these issues using discrete time modelling approach.
The rest of this paper is organized as follows: In section 2, a general description of discrete time SRGM based on NHPP is provided. Section 3 presents several NHPP based discrete time SRGMs and shows how these models can be integrated to form the resultant generalized model. Sections 4 and 5 provide the technique used for parameter estimation and the criteria used for validation of the models. The application of the presented models to actual fault-detection-data cited from real software development projects through data analysis and model comparisons are shown in section 6. We conclude this paper in section 7.

2 NHPP Based Discrete Time SRGMs: A General Description

During the software testing phase a software system is executed with a sample of test cases to remove software faults, which cause software failures [5]. A discrete counting process \([N(n); n \geq 0]\) is said to be an NHPP with mean value function \(m(n)\), if it satisfies the following conditions:

1. There are no failures experienced at \(n=0\), i.e., \(N(n=0)=0\).
2. The counting process has independent increments, that is, for any collection of the numbers of test cases \(n_1, n_2, \ldots, n_k\) where \((0<n_1<n_2<\ldots<n_k)\), the \(k\) random variables \(N(n_1), N(n_2)-N(n_1), \ldots, N(n_k)-N(n_{k-1})\) are statistically independent.
3. For any numbers of test cases \(n\) and \(n\), where \((0\leq n\leq n)\), we have
   \[
   \Pr[N(n)-N(n)=x]=\frac{[m(n)-m(n)]}{x!}\exp\{[m(n)-m(n)]\}, \quad x \geq 0
   \]
   The mean value function \(m(n)\) which bounded above and is non-decreasing in \(n\) represents the expected cumulative number of faults detected by \(n\) test cases. Then the NHPP model with \(m(n)\) is
   \[
   \Pr[N(n)=x]=\frac{[m(n)]}{x!}\exp\{[m(n)-m(n)]\}, \quad x \geq 0
   \]
   As a useful software reliability growth index, the fault detection rate per fault (per test case) after the \(n^{th}\) test case is given by
   \[
   q(n) = \frac{m(n+1)-m(n)}{m(\infty) - m(n)}, \quad n \geq 0
   \]
   where \(m(\infty)\) represents the expected number of faults to be eventually detected.

Let \(\bar{N}(n)\) denotes the number of faults remaining in the system after the \(n^{th}\) test case is given as
   \[
   \bar{N}(n) = N(\infty) - N(n)
   \]
   The expected value of \(\bar{N}(n)\) is given by:
   \[
   h(n) = m(\infty) - m(n)
   \]
   which is equivalent to the variance of \(N(n)\). Suppose that \(n_2\) faults have been detected by \(n\) test cases. The conditional distribution of \(N(n)\), given that \(N(n)=n_2\), is given by
   \[
   \Pr[N(n)=y|N(n)=n_2]=\frac{[E(n)]^y}{y!}\exp\{-[E(n)]\}
   \]
   which means a Poisson distribution with mean \(E(n)\), independent of \(n_2\). Now, the probability of no faults detected between the \(n^{th}\) and the \((n+h)^{th}\) test cases, given that \(n_2\) faults have been detected by \(n\) test cases, is given by:
   \[
   R(h|n)=\exp\{-[m(n+h)-m(n)]\}, \quad n, h \geq 0
   \]
   The above conditional reliability function \(R(h|n)\) called a software reliability function based upon a NHPP for a Discrete Time Model and is independent of \(n_2\).

Notations

- \(a\) Initial fault-content of the software.
- \(a(n)\) Over-all fault-content dependent on the number of test cases, which includes initial fault content and the number of faults introduced.
- \(b\) Fault detection rate per remaining faults per unit time.
- \(b(n+1)\) Fault detection rate dependent on the number of test cases.
- \(r\) Ratio of independent faults to the total number of faults in the software.
- \(m(n)\) Expected mean number of faults removal by the \(n^{th}\) test case.
- \(a\) Fault introduction rate per detected faults per test case.
- \(p\) The probability of fault introduction on a failure (i.e., the probability of perfect debugging).

3 Mathematical Modelling In Software Reliability

3.1 Exponential Model

This model is based on the assumption that the expected cumulative number of faults removed between the \(n^{th}\) and the \((n+1)^{th}\) test cases is proportional to the number of faults remaining after the execution of the \(n^{th}\) test case [5], satisfies the following difference equation:
   \[
   m(n+1)-m(n)=b(a-m(n))
   \]
   Solving the above difference equation using the probability generating function (PGF) with the initial conditions \(m(n=0)=0\), we get
   \[
   m(n)=a(1-(1-b)^n)
   \]
3.2 Imperfect Debugging Model
This model is based on the assumption that the debugging process is imperfect. The fault removal intensity satisfies the following difference equation,
\[ m(n + 1) - m(n) = b(ap - m(n)) \]  
(10)
Solving the above difference equation using the PGF with the initial conditions \( m(0) = 0 \), we get
\[ m(n) = a\left(1 - (1 - bp)^n\right) \]  
(11)
where \( m(0) = 0 \) and \( m(\infty) = a \).
If \( p = 1 \), the model reduces to the exponential model given in (9).

3.3 Flexible Model
This model is based on the assumption that the software contains two types of faults, namely mutually dependent and mutually independent. The mutually independent faults are those located on different execution paths of the software, therefore they are equally likely to be detected and removed. The mutually dependent faults are those faults located on the same execution path. According to the order of the software execution, some faults in the execution path will not be removed until their preceding faults are removed. The fault removal intensity satisfies the following difference equation,
\[ m(n + 1) - m(n) = b(n + 1)(a - m(n)) \]  
(12)
where \( b(n + 1) = b\left(r + (1 - r)\frac{m(n + 1)}{a}\right) \)  
(13)
By solving (5) using PGF with initial condition \( m(0) = 0 \), one can get the closed form solution as:
\[ m(n) = a\left(1 - (1 - bp)^{n+1}\right) \]  
(14)
where \( m(0) = 0 \) and \( m(\infty) = a \).
If \( r = 1 \), the model reduces to the imperfect debugging model given in (11). For different values of \( r \) different growth curves can be obtained and in that sense it is flexible.

3.4 Fault Generation Model
This proposed model is based on the assumption that fault introduction rate is a linear function of the number of test cases. The fault removal intensity satisfies the following difference equation,
\[ m(n + 1) - m(n) = b(a(n) - m(n)) \]  
(15)
where \( a(n) = a(1 + an) \)  
(16)
Solving the above difference equation using the PGF with the initial conditions \( m(0) = 0 \), we get
\[ m(n) = a\left[\left(1 - (1 - b)^n\right)\left(1 - \frac{\alpha}{b}\right) + an\right] \]  
(17)
where \( m(0) = 0 \) and \( m(\infty) = \infty \).

If \( \alpha = 0 \), the model reduces to the exponential model given in (9).

3.5 Generalized Model
Modifications to the flexible model in (14) have been proposed for increasing over-all fault-content [8,9]. The nature of over-all fault-content depends upon various factors like the skill of test team or testing strategy, number of test cases, and software size and complexity. Hence no single functional form can describe the growth in number of faults during testing phase and debugging process. This necessitates a modeling approach that can be modified without unnecessarily increasing the complexity of the resultant model. Here we show how this can be achieved for the flexible model by replacing the fault detection rate by a logistic function dependent on the number of executed test runs as proposed below,
\[ b(n + 1) = \frac{bp}{1 + \frac{1 - r}{r}(1 - bp)^{n+1}} \]  
(18)
An increasing \( a(n) \) implies an increasing total number of faults, and thus reflects fault generation. Whereas, \( b(n + 1) \) is a S-shaped curve that can capture the learning process of the software testers, and this function is affected by the probability of fault removal on a failure. The proposed generalized model satisfies the following difference equation:
\[ m(n + 1) - m(n) = b(n + 1)(a(n) - m(n)) \]  
(19)
By substituting (18) and (16) in (19) and then solving it using PGF with initial condition \( m(0) = 0 \), after tedious algebraic manipulations, one can get the closed form solution as given below.
\[ m(n) = \frac{a}{1 + \frac{1 - r}{r}(1 - bp)^n\left[1 - (1 - bp)^n\right]} \left(\frac{bp - \alpha}{bp}\right) + an \]  
(20)
where \( m(0) = 0 \) and \( m(\infty) = \infty \).
It should be pointed out here that the generalized model is more general than that of the flexible model because it includes the effect of fault generation, imperfect debugging, and has the fault generation (17), flexible (13), imperfect debugging (11) and the exponential (9) models as special cases.

4 Parameter Estimation Technique
The maximum likelihood estimation (MLE) method is used to estimate the unknown parameters of the developed models. Since all data sets used are given in the form of pairs \( (n_i, x_i) \), where \( x_i \) is the cumulative number of faults detected by \( n_i \) test cases \( (0 < n_1 < n_2 < \ldots < n_f) \) and \( n_i \) is
the accumulated number of test run executed to detect \( x_i \) faults. The likelihood function \( L \) for the unknown parameters with the superposed mean value function is given as

\[
L(\text{parameters}| (n_i, x_i)) = \prod_{i=1}^{f} \frac{[m(n_i) - m(n_{i-1})]^{{x_i-x_{i-1}}} \exp\left(-\left(m(n_i) - m(n_{i-1})\right)\right)}{(x_i - x_{i-1})!}
\]

(21)

The MLE of the SRGM parameters can be obtained by maximizing \( L \) in (21) w.r.t. the following constraints: \( a>0, 0<b<1, 0<p<1, 0<\rho<1, \alpha\geq0 \).

### 5 Model Validation and Comparison Criteria

#### 5.1 Model Validation

To check the validity of the proposed generalized model to describe the software reliability growth, it has been tested on three datasets (DS). The DS-I was for a radar system of size 124 KLOC after it was tested for 35 months in which 1301 faults were detected [14]. The DS-II was collected during 38 weeks of testing, 231 faults were detected [3].

#### 5.2 Comparison Criteria

The performance of an SRGM judged by its ability to fit the past software reliability data and to predict satisfactorily the future behavior from present and past data [1,3].

##### 5.2.1 Goodness of Fit Criteria

- **The Sum of Squared Error (SSE).** The distance of the model estimate value from the actual data is measured by the SSE as,

\[
SSE = \sum_{i=1}^{f} (\hat{m}(n_i) - x_i)^2
\]

(22)

where \( f \) is the number of observations. The lower value of SSE indicates less fitting error, thus better goodness of fit.

- **The Akaike Information Criterion (AIC).** The ability of the model to maximize the likelihood function that is directly related to the degrees of freedom during fitting is measured by the AIC as,

\[
AIC = -2\log(\text{max. likelihood function}) + 2N
\]

(23)

where \( k \) is the number of observations. Lower value of AIC indicates more confidence in the model thus a better fit and predictive validity.

- **Coefficient of Multiple Determination (\( R^2 \)).** 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,

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

(24)

\( R^2 \) 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.

#### 5.2.2 The 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 [1]. Let \( n_k \) be the last test case, \( x_k \) number of faults detected during the interval \((0,n_k]\), and \( \hat{m}(n_k) \) is the estimated value of the mean value function \( m(n) \) at \( n_k \), which is determined using the actually observed data up to an arbitrary test case \( n_j (0\leq n_j \leq n_k) \), in which \( n_j/n_k \) denotes the testing progress ratio. In other words, the number of failures by \( n_k \) can be predicted by the model 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 RPE. If the RPE value is negative/positive the model 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 \( \pm (10\%) \).

### 6 Data Analyses & Model Comparisons

#### 6.1 Goodness of Fit Analysis

Using MLE method, the estimated values of the proposed generalized model parameters for DS-I and DS-II are given in Table 1. According to the estimated values of the fault introduction rate parameter \( \alpha \) the fault detection process (i.e., the debugging process) in DS-I is perfect and no fault introduced during debugging, whereas in DS-II was not and the total number of faults introduced by the \( n^{th} \) test case is \( (\alpha(n=38)-\alpha) \), i.e., \( (315.64-145.98)=169.66\). According to the estimated values of the fault detection rate of additional faults parameter \( \rho \) the fault detection process does detect some of the remaining in DS-I, whereas in DS-II does not. That is may be the reason why the fault detection process resembles an S-shaped growth curve in DS-I while an Exponential growth curve in DS-II. The fitting of the proposed generalized model to DS-I and DS-II is graphically illustrated.

---

**Table 1:** Estimated values of the fault introduction rate parameter \( \alpha \) for DS-I and DS-II

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Estimated Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS-I</td>
<td>315.64</td>
</tr>
<tr>
<td>DS-II</td>
<td>231.98</td>
</tr>
</tbody>
</table>
in Figures 1 and 2. It may be interesting to note that the proposed generalized model reduces to the Flexible model and Imperfect Debugging model when applied to DS-I and DS-II respectively. Comparisons of the proposed generalized model and the well-documented Discrete Time Models in terms of goodness of fit is given in Tables 2 and 3 for DS-I and DS-II respectively. Note that the Exponential model fails to give any plausible estimation result when applied to DS-I. It is clearly seen from Table 3 that the proposed generalized model is the best among the models under comparison in terms of SSE, AIC, and R².

6.2 Predictive Validity Analysis
Both DS-I and DS-II are truncated into different proportions and used to estimate the parameters of the proposed generalized. For each truncation, one value of RPE ratio is obtained and is graphically illustrated in Figures 3 and 4. The RPE ratio of the proposed generalized model overestimates in DS-I and DS-II the fault detection process except for DS-II when the testing progress ratio is about 50% it underestimates the process. It is clearly seen from Figures 3 and 4 that 65% and 60% of the total test time are sufficient to predict the future of the fault detection process reasonably for DS-I and DS-II respectively.

7 Conclusions
In this paper we have presented several discrete time NHPP based SRGMs formulated under different sets of assumptions and testing environments, and have shown how these models can be integrated to form the resultant proposed generalized model. The resultant model is based on the assumptions that the numbers of fault detection attempts are more than actual fault-content but imperfect debugging does not change the content of faults in the software, and incorporated an S-shaped curve fault detection rate that can capture the learning-process of the software testers, and this function is affected by the probability of fault detection on a failure.

Based on our data analyses and model comparisons in Section 6, the resultant proposed generalized model not only fit the past fault-detection-count data sets well but also predict the future behavior of the fault-detection-process reasonably well. Finally, the proposed generalized model provides a large scope for further extensions and generalizations.

References
Tables and Figures

Table 1: Parameters Estimations (for DS-I and DS-II)

<table>
<thead>
<tr>
<th>Model</th>
<th>Data Set</th>
<th>Parameter Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>a</td>
</tr>
<tr>
<td>Proposed</td>
<td>DS-I</td>
<td>1331</td>
</tr>
<tr>
<td>Generalized</td>
<td>DS-II</td>
<td>146</td>
</tr>
</tbody>
</table>

Table 2: 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</td>
<td>b</td>
</tr>
<tr>
<td>Exponential [5]</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Proposed Generalized</td>
<td>See Table 1</td>
<td>7133</td>
</tr>
</tbody>
</table>

* indicates the model fails to give any plausible result

Table 3: 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</td>
<td>b</td>
</tr>
<tr>
<td>Exponential [5]</td>
<td>475.50</td>
<td>.0161</td>
</tr>
<tr>
<td>Delayed S-shaped [3]</td>
<td>234.23</td>
<td>.0919</td>
</tr>
<tr>
<td>Proposed Generalized</td>
<td>See Table 1</td>
<td>605</td>
</tr>
</tbody>
</table>

Goodness of Fit (for DS-I)

![Fig. 1](image1)

Goodness of Fit (for DS-II)

![Fig. 2](image2)

Predictive Validity (for DS-I)

![Fig. 4](image3)

Predictive Validity (for DS-II)

![Fig. 3](image4)