Discrete time modelling in software reliability engineering—a unified approach

Omar Shatnawi

Prince Hussein bin Abdullah Information Technology College, Al al-Bayt University, Jordan
E-mail: dromali@lycos.com

In the software reliability engineering literature, few attempts have been made to measure software reliability using discrete time modeling. One of the reasons can be attributed to the mathematical complexity involved in constructing such models. The proposed unified modeling approach provides a broad framework for developing NHPP type of discrete SRGMs. The framework adopts the number of test occasions/cases as a unit of fault detection/removal period, which is countable and more appropriate measure than CPU time or calendar time used in continuous SRGMs. And classifies the faults that encountered during software testing phase into three types of faults namely, simple, hard and complex according to their removal complexity. Accordingly, their fault removal processes is modelled separately and the total fault removal phenomena is the superposition of the underlying processes. Such type of modelling approach is very much suited for object-oriented and distributed systems development environments. Actual software reliability data have been used to demonstrate the proposed framework.

Keywords: Software reliability engineering, software testing, NHPP, SRGMs, test cases, fault severity.

1. INTRODUCTION

Successful operation of any computer system depends largely on its software components. Thus, it is very important to ensure the quality of the underlying software in the sense that it performs its functions that it is designed and built for. To express the quality of the software system to the end users, some objective attributes such as reliability and availability should be measured. Software reliability is the most dynamic quality attribute (metric), which can measure and predict the operational quality of the software system. Software reliability model (SRM) is the tool, which can be used to evaluate the software quantitatively, develop test cases, schedule status and monitor the change in reliability performance. In particular, SRMs that describe software failure occurrence or fault removal phenomenon in the system testing phase are called software reliability growth models (SRGMs). Among others, non-homogeneous Poisson process (NHPP) models can be easily applied in the actual software development. Generally, the SRGMs are classified into two groups. The first group contains models, which adopt the CPU or calendar time as a unit of the fault removal period. Such models are called continuous time models. The second group contains models, which adopt the number of test cases as a unit of the fault removal period. Such models are called discrete time models, since the unit of the fault removal period is countable. A test case can be a single computer test run executed in an hour, day, week or even month [1-3]. 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 there are fewer in the second group.

The remainder of this paper is organized as follows. Section 2 presents the framework. Section 3 defines some software reliability evaluation measures. Section 4 and 5 provide the method used for parameter estimation and the criteria used for validation and evaluation of the developed framework respectively. The applications of the framework to actual software reliability data collected from real software development projects are shown in Section 6. Section 7 concludes this paper.
2. DISCRETE TIME MODELLING IN SOFTWARE RELIABILITY ENGINEERING

2.1 Framework development

In the software reliability engineering literature, most researchers assume a constant fault removal rate per fault in deriving their SRGM [2-8]. That is, they assume that all faults have equal probability of being removed during the software testing process, and the rate remains constant. In reality, the fault removal rate strongly depends on the skill of test teams, program size and software testability. Through real data experiments and analyzes on several software development projects, it has been observed the fault removal rate has three possible trends as time progresses: increasing, decreasing or constants [3, 5, 9-14]. They also assume that the faults in the software are of the same type. Such assumption helps to simplify the problem of modelling and provides to a certain extent plausible results. However, this assumption is not truly representative of reality. It has been observed that the software contains different types of faults and each fault requires different strategies and different amount of testing effort to remove it. If this assumption is not taken into account, the SRGM may give misleading results. To counter this problem several SRGM have been developed [3, 15, 17, 18]. In these models, they ignore the role of the learning-process during the testing phase by not accounting for the experience gained with the progress of testing.

To address these issues a framework is developed through a unified modeling approach, which incorporates a logistic learning function during the removal phase, for capturing variability in the growth curves depending on the environment it is being used and at the same time it has the capability to reduce either to exponential or S-shaped growth curves.

2.2 Framework assumptions and notations

2.2.1 Framework assumptions

1. Fault removal phenomenon follows the NHPP.
2. Software is subject to failures during execution caused by the remaining faults.
3. Software contains three types of faults namely; simple, hard, and complex modelled by a one-stage, two-stage and three-stage removal processes respectively.
4. Each time a failure occurs, an immediate (delayed) effort takes place to decide the cause of the failure in order to remove it. The time delay between the failure observation and its subsequent represents the severity of the faults.
5. Fault removal complexity is proportional to the amount of testing-effort required to remove the fault. The testing-effort expenditures are represented by the number of stages required to remove the fault after the failure observation / fault isolation (with time delay between the stages).
6. The fault removal rate is a discrete logistic function, which is a non-decreasing S-shaped curve, it is expected the learning-process of the test team will grow with the number of test run executed.
7. 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 run,

2.2.2 Framework notations

- $a_i$ Fault-content of type $i$ ($\sum_{i=0}^{3} a_i =a$), where $a$ is the total fault-content.
- $b_i$ Proportionality constant failure rate / fault isolation rate per fault of type $i$.
- $m_i f(n)$ Logistic learning function, i.e., fault removal rate per fault of type $i$.
- $m_i q(n)$ Mean number of failures caused by fault-type $i$ by $n$ test cases.
- $m_i r(n)$ Mean number of fault isolated of fault-type $i$ by $n$ test cases.
- $m_{ir}(n)$ Mean number of fault removed of fault-type $i$ by $n$ test cases.
- $\beta$ Constant parameter in the logistic function.

2.3 Framework formulation

2.3.1 Modelling the simple faults (i.e., type 1)

The simple fault removal phenomenon is modelled as a one-stage process

\[ m_{1r}(n+1) - m_{1r}(n) = b_1 (a_1 - m_{1r}(n)) \]  

Solving the above difference equation using the probability generating function (PGF) with the initial condition $m_{1r}(n = 0) = 0$, we get

\[ m_{1r}(n) = a_1 (1 - (1 - b_1)^n) \]  

2.3.2 Modelling the hard faults (i.e., type 2)

The fault removal phenomenon is modelled as a two-stage process,

\[ m_{2f}(n+1) - m_{2f}(n) = b_2 (a_2 - m_{2f}(n)) \]  

\[ m_{2r}(n+1) - m_{2r}(n) = b_2 (n+1) (m_{2f}(n+1) - m_{2f}(n)) \]  

where

\[ b_2(n+1) = \frac{b_2}{1 + \beta (1 - b_2)^{n+1}} \]  

which is a discrete version of the logistic function proposed in the literature [12]. It accounts for the learning-process phenomenon of the test team. (see assumption 6)

Solving the above system of difference equations using the PGF with the initial conditions $m_{2f}(n = 0) = 0$ and $m_{2r}(n = 0) = 0$, we get

\[ m_{2r}(n) = a_2 \frac{1 - (1 + b_2 n)(1 - b_2)^n}{1 + \beta (1 - b_2)^n} \].  

2.3.3 Modelling the complex faults (i.e., type 3)

The complex fault removal phenomenon is modelled as a three-stage process,

\[ m_{3f}(n + 1) - m_{3f}(n) = b_3 \left( a_3 - m_{3f}(n) \right) \]  

(6)

\[ m_{3f}(n + 1) - m_{3f}(n) = b_3 (m_{f}(n + 1) - m_{3f}(n)) \]  

(7)

\[ m_{3f}(n + 1) - m_{3f}(n) = b_3 (n + 1) \left( m_{3f}(n + 1) - m_{3f}(n) \right) \]  

(8)

where

\[ b_3(n + 1) = \frac{b_3}{1 + \beta(1 - b_3)^{n+1}}. \]

Solving the above system of difference equations using the PGF with the initial conditions \( m_{3f}(n = 0) = 0 \), \( m_{3f}(n = 0) = 0 \) and \( m_{3f}(n = 0) = 0 \), we get

\[ m_{3f}(n) = a_3 \left( 1 - (1 + b_3 n)^{-1} \right)(1 - b_3)^n. \]  

(9)

2.3.4 Modelling the total fault removal phenomena

The developed framework is the superposition of the NHPP with mean value functions given in equations (2), (5) and (9). Thus, the mean value function of the superposed NHPP is

\[ m_{GF-k}(n) = \sum_{i=1}^{3} m_{ir}(n) \]

\[ = a_1 \left( 1 - (1 - b_1)^n \right) \]

\[ + \sum_{i=2}^{3} a_i \left( 1 - \frac{1 + \frac{j(n+1)}{j(n+1)}}{(1 + \beta(1 - b_1)^n)} \right)(1 - b_i)^n \]  

(10)

where \( m_{GF-k}(n) \) provides the framework with \( k(= 1, 2, 3) \) types of faults and \( j \) is number of stages required to remove the fault after its failure observation / fault detection and is dependent on the type of the fault. The fault removal rate per fault for the three fault-type are given respectively as,

\[ d_1(n) = \frac{m_1(n + 1) - m_1(n)}{a_1 - m_1(n)} = b_1 \]  

(11)

\[ d_2(n) = \frac{m_2(n + 1) - m_2(n)}{a_2 - m_2(n)} \]

\[ = \frac{b_2(1 + \beta + b_2 n) - b_2(1 + \beta(1 - b_2)^n)}{(1 + \beta(1 - b_2)^n)(1 + \beta + b_2 n)} \]  

(12)

\[ d_3(n) = \frac{m_3(n + 1) - m_3(n)}{a_3 - m_3(n)} \]

\[ = \frac{b_3 \left( 1 + \beta + b_3 n + \frac{b_3^2(n+1)}{2} \right) - b_3(1 + \beta(1 - b_3)^n)(1 + b_3 n)}{(1 + \beta(1 - b_3)^n)(1 + \beta + b_3 n + \frac{b_3^2(n+1)}{2})} \]  

(13)

It is observed that \( d_1(n) \) is constant with respect to \( n \), while \( d_2(n) \) and \( d_3(n) \) increase monotonically with \( n \) and tend to constants \( b_2 \) and \( b_3 \) as \( n \to \infty \). Thus, in the steady state, \( m_{2r}(n) \) and \( m_{3r}(n) \) behave similarly as \( m_{1r}(n) \) and hence there is no loss of generality in assuming the steady state rates \( b_2 \) and \( b_3 \) to be equal to \( b_1 \). After substituting \( b_2 = b_3 = b_1 \) in the right hand side of equations (12) and (13), one can see that \( d_1(n) > d_2(n) > d_3(n) \), which is in accordance with the severity of the faults \([3, 19]\).

Let \( b_1 = b_2 = b_3 = b \) (say). Then we may write equation (10) as follows

\[ m_{F-k}(n) = \sum_{i=1}^{3} m_{ir}(n) = a_1 \left( 1 - (1 - b)^n \right) \]

\[ + \sum_{i=2}^{3} a_i \frac{1 - \left( \frac{1 + \frac{j(n+1)}{j(n+1)}}{(1 + \beta(1 - b)^n)} \right)(1 - b)^n}{(1 + \beta(1 - b)^n)} \]  

(14)

The framework so defined in equation (14) is very interesting from various points of view. Besides the above-mentioned interpretation, it can also reduce to the exponential \([2]\), delay S-shaped and generalized Erlang models of \([3]\). Therefore, it is able to model both cases of strictly decreasing failure intensity and the case of increasing-and-decreasing failure intensity.

2.4 Remarks on the framework

It may be interesting to point out here that if the fault removal rate of the simple fault is assumed to be a logistic function. Then equation (2) is defined as

\[ m_{ir}(n) = a_1 \frac{1 - (1 - b_1)^n}{1 + \beta(1 - b_1)^n}. \]  

(15)

The analysis followed in Section (2.3.5) can be again applied. Accordingly, the proposed framework so defined in equation (10) can be rewritten as

\[ m_{F-k}(n) = \sum_{i=1}^{3} m_{ir}(n) \]

\[ = \sum_{i=1}^{3} a_i \frac{1 - \left( \frac{1 + \frac{j(n+1)}{j(n+1)}}{(1 + \beta(1 - b_1)^n)} \right)(1 - b)^n}{(1 + \beta(1 - b)^n)}. \]  

(16)

This version of the developed framework can reduce to the discrete versions of most of the existing NHPP models of \([3, 6, 9, 11, 15, 18]\).

3. SOFTWARE RELIABILITY MEASUREMENTS

Let \( [N(n); n \geq 0] \) denotes a discrete counting process representing the cumulative number of failures experience or fault removed by \( n \) test cases, then it can normally be modeled as a \( n \) NHPP with the superposed mean value function \( m(n) \). The NHPP model with \( m(n) \) is formulated by

\[ \Pr\{N(n) = x_f\} = \frac{(m(n))^{x_f}}{x_f!} \exp[-m(n)], \quad x_f \geq 0 \]  

(17)
Based on the NHPP model with \( m(n) \), the following quantitative reliability measures can be derived.

### 3.1 Expected number of remaining faults

Let \( W(n) \) denotes the number of faults remaining in the software after the execution of the \( n^{th} \) test case, then we have

\[
W(n) = N(\infty) - N(n).
\]

The expected value of \( W(n) \) is given by

\[
E[W(n)] = m(\infty) - m(n) = H(n)
\]

which is equivalent to the variance of \( W(n) \), where \( m(\infty) \) represents the expected number of faults to be eventually removed.

### 3.2 Software reliability

The probability of no failures (faults) experienced (removed) between the \( n^{th} \) and \( (n+s)^{th} \) test cases where \( s \) is the mission test case, given that \( x_o \) failures (faults) have been experienced (removed) by \( n \) test cases, is given by

\[
R(s|n) = \exp(m(n) - m(n + s)), \ s \geq 0
\]

which means a reliability function in \( n \), independent of \( x_o \). This is called a conditional reliability function.

### 4. PARAMETER ESTIMATION TECHNIQUE

The maximum likelihood estimation (MLE) method is used to estimate the unknown parameters of the developed framework. Since all data sets used are given in the form of pairs \( (n_i, x_i) \) \( i = 1, 2, \ldots, f \), 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

\[
L(\text{parameters}|n_i, x_i) = \prod_{i=1}^{f} \frac{[m(n_i) - m(n_{i-1})]^{x_i-x_{i-1}}}{(x_i-x_{i-1})!} \times \exp \left( -\frac{m(n_i) - m(n_{i-1})}{2} \right)
\]

Taking natural logarithm of equation (21) we get

\[
\ln L = \sum_{i=1}^{f} (x_i-x_{i-1}) \ln \left[ m(n_i) - m(n_{i-1}) \right] - \left( m(n_1) - m(n_{f-1}) \right) - \sum_{i=1}^{f} \ln \left( (x_i-x_{i-1})! \right)
\]

The MLE of the SRGM parameters can be obtained by maximizing \( L \) in equation (21) or (22) with respect to the following constraints: \( a > 0, 0 < b < 1, \beta \geq 0 \).

### 5. FRAMEWORK VALIDATION AND COMPARISON CRITERIA

#### 5.1 Framework validation

To check the validity of the developed framework to describe the software reliability growth, it has been tested on two actual software reliability data sets cited from real software development projects [5]. The first data set (DS-I) had been collected during 20 weeks of testing one of four major releases of the software products at Tandem Computers Company, Los Angeles (CA), 100 faults were detected during the period. The second data set (DS-II) had been collected during 111 days of test operations performed in a day are regarded to be a test instance of a Real-Time Control software system consists of about 200 modules having, on an average, 1K LOC of a high-level language such as Fortran, 481 faults were detected during the period.

#### 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 [3, 8].

##### 5.2.1 Goodness of fit criteria

1. **The Sum of Squared Error (SSE)**. The difference between the simulated data \( m(n_i) \) and the observed (reported) data \( x_i \) is measured by the SSE as,

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

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

2. **The Akaike Information Criterion (AIC)**. This criterion was first proposed as SRGM model selection tool [20]. It is defined as,

\[
AIC = -2(\text{value of max. log likelihood function}) + 2(\text{number of parameters used in the model})
\]

Lower value of AIC indicates more confidence in the model thus a better fit and predictive validity.

3. **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,

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

\( R^2 \) 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 [8].

74
5.2.2 Predictive validity criterion

This criterion was proposed by [8]. The relative prediction fault (RPF) is defined as,

\[
\text{RPF} = \frac{\hat{m}(n_f) - x_f}{x_f}
\]

where \(x_f\) is the cumulative number of faults removed after the execution of the last test run \(n_f\) and \(m(n_f)\), is the estimated value of the SRGM \(m(n_f)\), which determined using the actually observed data up to an arbitrary test case \(n_e(\leq n_f)\). If the RPF value is negative / positive the model is said to underestimate / overestimate the fault removal process. A value close to zero indicates more accurate prediction, thus more confidence in the model. The value is said to be acceptable if it is within \((\pm 10\%)\) [3].

6. DATA ANALYSES AND MODEL COMPARISONS

The models under comparison are \(m_{F-1}(n)\), \(m_{F-2}(n)\) and \(m_{F-3}(n)\) that estimates the presence of one, two and three types of faults respectively.

6.1 Goodness of fit analysis

Table 1 demonstrates the results of the parameter estimation and the comparison criteria in terms of goodness of fit of the models under comparison for DS-I. On applying \(m_{F-2}(n)\), the model reveals the presence of two types of faults where the majority is of type 2. Accordingly, the fault removal phenomena are described by a combination of faults types 1 and 2. On applying \(m_{F-3}(n)\), the model reveals the presence of only two types 1 and 3 of faults where the majority is of type 3. Accordingly, the fault removal phenomena are described by a combination of faults types 1 and 3. Besides, it is clearly seen that the results of the goodness of fit metrics of the models are improving with the introduction of more types of faults.

Table 2 demonstrates the results of the parameter estimation and the comparison criteria in terms of goodness of fit of the models under comparison for DS-II. On applying \(m_{F-3}(n)\), the model reduces to \(m_{F-2}(n)\), which reveals the presence of only two types 1 and 2 of faults where the majority of faults are of type 2. Accordingly, the fault removal phenomena are described by a combination of faults types 1 and 2. Besides, it is clearly seen that the results of the goodness of fit metrics of the models are improving with the introduction of more types of faults.

Figures 1 and 2 graphically illustrate the fitting of the models under comparison to DS-I and DS-II. It is clearly seen from the figure that the models fit both of them excellently. Among the models under comparison, the models (i.e., \(m_{F-2}(n)\) and \(m_{F-3}(n)\)) that consider the impact (severity) of the fault-type (complexity) are better descriptive than the model (i.e., \(m_{F-1}(n)\)) that does not consider that.

6.2 Predictive validity analysis

Both DS-I and DS-II are truncated into different proportions and used to estimate the parameters of the models under comparison. For each truncation, one value of predictive validity metric (RPF in percent) is obtained. It is observed that the predictive validity varies from one truncation to another. Figure 3 graphically illustrates the results of the (RPF\%) of the models under comparison for DS-I. On applying \(m_{F-1}(n)\), it is observed that the model overestimates the fault removal process. On applying \(m_{F-2}(n)\) and \(m_{F-3}(n)\), it is observed that the models underestimate the fault removal process except for the truncations 85%, 95% and 100% they overestimate the process. Figure 4 graphically illustrates the results of the (RPF\%) of the models under comparison for DS-II. On applying the models, it is observed that all the models overestimate the fault removal process for all the truncations. Among the models under comparison, the models (i.e., \(m_{F-2}(n)\) and \(m_{F-3}(n)\)) that consider the impact of the fault-type are better predictive than the model (i.e., \(m_{F-1}(n)\)) that does not consider that.
Table 1 Parameter estimation & goodness-of-fit metrics results 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>$a_1$</td>
<td>$a_2$</td>
</tr>
<tr>
<td>$m_{F-1}(n)$</td>
<td>130.20</td>
<td>—</td>
</tr>
<tr>
<td>$m_{F-2}(n)$</td>
<td>38.14</td>
<td>62.90</td>
</tr>
<tr>
<td>$m_{F-3}(n)$</td>
<td>39.26</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 2 Parameter estimation & goodness-of-fit metrics results 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>$a_1$</td>
<td>$a_2$</td>
</tr>
<tr>
<td>$m_{F-1}(n)$</td>
<td>538.10</td>
<td>—</td>
</tr>
<tr>
<td>$m_{F-2}(n)$</td>
<td>130.88</td>
<td>351.67</td>
</tr>
<tr>
<td>$m_{F-3}(n)$</td>
<td>130.88</td>
<td>351.67</td>
</tr>
</tbody>
</table>

6.3 Software reliability measurements analysis

Both DS-I and DS-II are used to graphically illustrate the software evaluation measures based on the models under comparison. Figures 5 and 6 explain the fitting of the models to the actual remaining number of faults to DS-I and DS-II respectively. Figures 7 and 8 explain the software reliability growth when the mission test case (i.e., $x = 1$) for DS-I and DS-II. Among the models under comparison, the models (i.e., $m_{F-2}(n)$ and $m_{F-3}(n)$) that consider the impact of the fault-type have shown better results compared with the than the model (i.e., $m_{F-1}(n)$) that does not consider that.

7. CONCLUSIONS

As the software technology is changing with the passage of time, the modelling approach should be addressed accordingly. We feel that the proposed unified modelling approach, which provides the developed framework, is a step in that direction. Based on Section 6, it is observed that the introduction of more fault types increases the flexibility of the model and thus its retrodictive and predictive capability increases. To increase the flexibility of the model, the model should include all the types of faults that exist in the software. As the number of fault types may tend to be large, modelling each fault type individually is not
practically possible. However, if the faults are classified into a smaller number of groups, where each group contains the faults of common characteristics, the number of faults can be reduced to the level that it can be practically handled by the developed framework. Finally, the developed framework provides a large scope for further extension and generalization.

REFERENCES
