Unified framework for discrete software reliability growth modeling with change point and a related release time problem

P. K. Kapur\textsuperscript{1}, Abhishek Tandon\textsuperscript{1} and Ravi Kumar\textsuperscript{1}

\textsuperscript{1} Department of Operational Research, University of Delhi, Delhi-110007, India
E-mail: pkkapur1@gmail.com

accepted July 17, 2010

Summary

In this paper we develop a unified framework for modeling discrete software reliability growth modeling with change point using various distributions. The modeling framework is developed using the probability generating function before and after the change point. Change point can be defined as the point where the fault detection/isolation/correction rate changes due to variations in testing skills, testing strategy and types of faults. Further in this paper we develop a software release policy in which we establish the optimum number of test cases which minimize the expected total cost subject to a specified level of reliability. The proposed models are validated on two data sets and the release policy is justified with a numerical example.

Key words: Software reliability, Software reliability growth model (SRGM), Non-homogeneous Poisson process (NHPP), test occasions (cases), release time.

1. INTRODUCTION

There has been a volatile growth in software industry for past 25 years. The radical progress in the computer technology have greatly influenced the production of large-scale computer systems and posed several challenges to the management of software predicament. The high development cost, delayed delivery, lack of systematic design techniques and unreliable software to name a few. In the early seventies, the software engineering discipline emerged to establish and use sound engineering principles in order to economically obtain software systems that are not only reliable but also work efficiently on real machines, thus bringing the software development under the engineering umbrella. The immediate concern of software engineering was aimed at developing highly reliable software, scheduling and systematizing the software development process at reduced costs.
Software reliability measurement and prediction is important to assess the performance of software system. The reliability of the software is quantified in terms of the estimated number of faults remaining in the software system. During the testing phase, the emphasis is on reducing the remaining fault content hence increasing the software reliability. The tool used to estimate the fault content, failure rate and fault removal rate per fault in software and to predict the reliability of the software at the release time are Software Reliability Growth Model (SRGMs). However no software can be tested indefinitely in order to make it bug free owing to the prevailing paradox that software users requirements are conflicting with the developers. Software users want faster deliveries; cheaper software as well as quality product whereas software developers want to minimize their development cost, maximize the profit margins and meet the competitive requirements. Hence an imperative decision problem is to determine when to stop testing and release the software system to the user known as “Software Release Time Problem”.

Most of these SRGMs are NHPP based and utilize historical failure data collected during the testing phase to evaluate the quality of software. 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 as a unit of fault detection period is known as continuous time models [2,3,5] proposed under different set of assumptions. These models describe both exponential and S-shaped relationship between time/effort and the corresponding number of faults removed. The second group contains models, which use the test cases as a unit of fault detection/removal period. These models are called discrete time models, since the unit of software fault removal period is countable [4,]. 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.

The utility of discrete SRGMs cannot be underestimated in spite of difficulties in terms of mathematical complexity involved. Discrete models are proposed regularly as the software failure data sets are discrete, and these models many times provide better fit than their continuous time counterparts. Yamada et al. [16-18] proposed exponential and modified exponential discrete SRGM assuming 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 run. Further we incorporate the concept of change point in discrete modeling. Change point is referred to a point where the fault detection rate changes due to various reasons such as change in the testing personal, the severity of faults etc. In discrete software reliability growth modeling, after a finite number of test cases, the detection rate is changed and the concept of change point is drawn in.Kapur et al [13] proposed the concept of change point in discrete software reliability growth modeling.

2. DISCRETE SOFTWARE RELIABILITY GROWTH MODELING

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 $[N(n), n \geq 0], (n = 0,1,2,...)$ 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 $m(n)$ 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:

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

(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:

\[
\Pr[N(n) = x] = \frac{[m(n)]^x}{x!} \exp[-[m(n)]].
\]

(2)

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

\[
\bar{N}(n) = N(\infty) - N(n).
\]

(3)

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

\[
E(n) = m(\infty) - m(n).
\]

(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 \( \bar{N}(n) \), given that \( N(n) = n_d \), is given by:

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

(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:

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

(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 \( n_d \).

### 3. MODEL DEVELOPMENT

In this paper we propose a discrete generalized modeling framework proposing several models based on various probability distributions. The models are based on following assumptions:

1. Failure fault removal phenomenon is modeled by NHPP.
2. Software is subject to failures during execution caused by faults remaining in the software.
3. Failure rate is equally affected by all the faults remaining in the software.
4. Fault detection / removal rate may change at any test case.

**Notations**

- \( a \): Expected number of faults in the software
- \( b_1 \): Detection rate before change point
- \( b_2 \): Detection rate after change point
\( m(n) \): Number of faults removed by \( n \) test cases

\( n_1 \): Change point

\( N \): Total test cases

Thus under the common assumptions for software reliability growth modeling, we consider the following differential difference equation.

\[
m(n+1) - m(n) = \begin{cases} 
\frac{f_1(n)}{1-F_1(n)} \left( a - m(n) \right) & 0 \leq n \leq n_1 \\
\frac{f_2(n)}{1-F_2(n)} \left( a - m(n) \right) & n \geq n_1 
\end{cases}
\]  

(7)

Where, \( \frac{f_1(n)}{1-F_1(n)} \) and \( \frac{f_2(n)}{1-F_2(n)} \) is defined as the generalized hazard rate before and after change point.

Whereas the mean value function \( m(n) \) is derived as:

\[
m(n) = \begin{cases} 
a F_1(n) & \text{for } 0 \leq n \leq n_1 \\
\frac{a(1-(1-F_1(n))(1-F_2(n)))}{(1-F_2(n))} & \text{for } n > n_1 
\end{cases}
\]

3.1 Derivation of various SRGMs

The fault detection rate is generalized by a variety of Failure-occurrence time distributions which are

**SRGM-1**

\[
F(n) = \begin{cases} 
(1-(1-b_1)^n) & \text{for } 0 \leq n \leq n_1 \\
(1-(1-b_2)^n) & \text{for } n > n_1 
\end{cases}
\]

\[
m(n) = \begin{cases} 
a(1-(1-b_1)^n) & \text{for } 0 \leq n \leq n_1 \\
a(1-(1-b_1)^n)(1-(1-b_2)^n) & \text{for } n > n_1 
\end{cases}
\]

(9)

**SRGM-2**

\[
F(n) = \begin{cases} 
(1-(1+b_1 n)(1-b_1)^n) & \text{for } 0 \leq n \leq n_1 \\
(1-(1+b_2 n)(1-b_2)^n) & \text{for } n > n_1 
\end{cases}
\]

\[
m(n) = \begin{cases} 
\frac{a(1-(1+b_1 n)(1-b_1)^n)}{a(1-(1+b_2 n)(1-b_2)^n)} & \text{for } 0 \leq n \leq n_1 \\
\frac{a(1-(1+b_1 n)(1-b_1)^n)(1-(1-b_2)^n)}{a(1-(1+b_2 n)(1-b_2)^n)(1-(1-b_2)^n)} & \text{for } n > n_1 
\end{cases}
\]

(10)
The above model is discussed by Kapur et al [13].

**SRGM-3**

\[
F(n) = \begin{cases} 
\frac{b_1}{1+\beta(1-b_1)^n} & \text{for } 0 \leq n \leq n_1 \\
\frac{b_2}{1+\beta(1-b_2)^n} & \text{for } n > n_1 
\end{cases}
\]

In the above logistic rate function it is assumed that \( \beta \) remains same before and after change-point for simplicity.

The corresponding mean value function \( m(t) \) is given by:

\[
m(n) = \begin{cases} 
\frac{1-\beta}{1+\beta(1-b_1)^n} & \text{for } 0 \leq n \leq n_1 \\
\frac{1-\beta(1-b_1)^n}{1+\beta(1-b_2)^n(1-b_2)^{n_1-n_0}} & \text{for } n > n_1
\end{cases}
\]

**SRGM-4**

\[
F(n) = \begin{cases} 
(1-(1-h_1)^n)^k & \text{for } 0 \leq n \leq n_1 \\
(1-(1-h_2)^n)^k & \text{for } n > n_1
\end{cases}
\]

where \( k > 0 \) is the shape parameter.

\[
m(n) = \begin{cases} 
(1-(1-h_1)^n)^k & \text{for } 0 \leq n \leq n_1 \\
(1-(1-h_2)^n)^k(1-b_2)^{n_1-n_0} & \text{for } n > n_1
\end{cases}
\]

**4. MODEL VALIDATION, PARAMETER ESTIMATION AND COMPARISON CRITERIA**

To illustrate the estimation procedure and application of the SRGM (existing as well as proposed) we have carried out the data analysis of real software data set. The parameters of the models have been estimated using statistical package SPSS and the change-point of the data sets have been judged by using change-point analyzer.

**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 [34]. The change-point for this data set is \( 17^{th} \) month.

**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 [31]. The change-point for this data set is \( 6^{th} \) week.
4.1 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.

4.2 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”?

4.2.1 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.

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

Where $k$ is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit [18].

4.2.2 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.

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

$R^2$ 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 [18].

4.2.3 Bias

The difference between the observation and prediction of number of failures at any instant of time $i$ is known as PE$_i$ (prediction error). The average of PEs is known as bias. Lower the value of Bias better is the goodness of fit [20].

4.2.4 Variation

The standard deviation of prediction error is known as variation.

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

Lower the value of Variation better is the goodness of fit [20].

4.2.5 Root Mean Square Prediction Error

It is a measure of closeness with which a model predicts the observation.

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

Lower the value of Root Mean Square Prediction Error better is the goodness of fit [20].
### 4.3 Data Analyses

#### For DS-1

The parameter estimation and comparison criteria results for DS-1 of all the models under consideration can be viewed through Table 1 (a) and Table 1 (b). It is clear from the table that the value of $R^2$ for SRGM-3 and SRGM-5 are higher and value of MSE, Bias, Variation and RMSPE are lower in comparison with other models and provides better goodness of fit for DS-1.

#### For DS-2

The parameter estimation and comparison criteria results for DS-2 of all the models under consideration can be viewed through Table 2 (a) and Table 2 (b). It is clear from the table that the value of $R^2$ for SRGM-3 and SRGM-5 are higher and value of MSE, Bias, Variation and RMSPE are lower in comparison with other models and provides better goodness of fit for.

#### Table 1 (a). Model Parameter Estimation Results (DS-1)

<table>
<thead>
<tr>
<th>Models</th>
<th>$a$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$k$</th>
<th>$\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM-1</td>
<td>2705</td>
<td>0.019</td>
<td>0.023</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SRGM-2</td>
<td>1639</td>
<td>0.092</td>
<td>0.095</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SRGM-3</td>
<td>1321</td>
<td>0.213</td>
<td>0.211</td>
<td>25</td>
<td>-</td>
</tr>
<tr>
<td>SRGM-4</td>
<td>1303</td>
<td>0.0007</td>
<td>0.0007</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

#### Table 1 (b). Model Comparison Results (DS-1)

<table>
<thead>
<tr>
<th>Models</th>
<th>$R^2$</th>
<th>MSE</th>
<th>Bias</th>
<th>Variation</th>
<th>RMSPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM-1</td>
<td>0.97367</td>
<td>5608.09</td>
<td>-1.28E-05</td>
<td>75.9804</td>
<td>75.9804</td>
</tr>
<tr>
<td>SRGM-2</td>
<td>0.98833</td>
<td>2486.007</td>
<td>-3.89E-06</td>
<td>50.5877</td>
<td>50.5877</td>
</tr>
<tr>
<td>SRGM-3</td>
<td>0.99930</td>
<td>149.5196</td>
<td>-4.96E-07</td>
<td>12.4063</td>
<td>12.4063</td>
</tr>
<tr>
<td>SRGM-4</td>
<td>0.99872</td>
<td>271.789</td>
<td>-22.08</td>
<td>25.0093</td>
<td>33.3660</td>
</tr>
</tbody>
</table>

#### Table 2 (a). Model parameter estimation results (DS-2)

<table>
<thead>
<tr>
<th>Models</th>
<th>$a$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$k$</th>
<th>$\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM-1</td>
<td>614</td>
<td>0.040</td>
<td>0.044</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SRGM-2</td>
<td>401</td>
<td>0.189</td>
<td>0.169</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SRGM-3</td>
<td>365</td>
<td>0.232</td>
<td>0.217</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>SRGM-4</td>
<td>372</td>
<td>0.022</td>
<td>0.018</td>
<td>1.625</td>
<td>-</td>
</tr>
</tbody>
</table>

#### Table 2 (b). Model comparison results (DS-2)

<table>
<thead>
<tr>
<th>Models</th>
<th>$R^2$</th>
<th>MSE</th>
<th>Bias</th>
<th>Variation</th>
<th>RMSPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM-1</td>
<td>0.98812</td>
<td>122.6662</td>
<td>-6.55E-07</td>
<td>11.3789</td>
<td>11.3789</td>
</tr>
<tr>
<td>SRGM-2</td>
<td>0.99054</td>
<td>97.61703</td>
<td>5.32E-06</td>
<td>10.1508</td>
<td>10.1508</td>
</tr>
<tr>
<td>SRGM-3</td>
<td>0.99270</td>
<td>75.29769</td>
<td>3.40E-06</td>
<td>8.9152</td>
<td>8.9152</td>
</tr>
<tr>
<td>SRGM-4</td>
<td>0.99143</td>
<td>88.41156</td>
<td>-2.85E-05</td>
<td>9.6603</td>
<td>9.6603</td>
</tr>
</tbody>
</table>

92
5. RELEASE PROBLEM

A lot of work has been done on the release time problems in software reliability in the literature. Release time problems are of great importance for the developer of the software as longer the testing phase higher is the reliability and the smaller the operational cost in terms of warranty and risk costs. However, delays in software release increase testing/debugging and penalty costs. Hence, determination of optimum no of test cases with a specified level of reliability is of enormous significance. In this section we discuss an optimal software release time policy under the assumption that the fault removal rate is altered by the introduction of new testing tools and/or techniques after a fixed number of test cases. We further determine the optimal number of test cases before releasing the software.

Now we define some additional parameters associated with the release policy under discussion:

\[ N^* : \text{optimal number of test occasions (cases) before the release of the software system} \]

\[ n_1 : \text{change-point} \]

\[ C_1 : \text{Cost of removing one fault before the change point in the testing phase} \]

\[ C_2 : \text{Cost of removing one fault after the change point in the testing phase} \]

\[ C_3 : \text{Cost of removing one fault in the operational phase} \]

\[ C_4 : \text{Cost per unit testing time} \]

\[ R_o : \text{Aspiration level for the software reliability} \]

The expected software cost function for the testing and operational phase can be written as:

\[
C(N) = C_1 m_1(n_1) + C_2 [m_2(N) - m_1(n_1)] + C_3 [a - m_2(N)] + C_4(N).
\]

Now,

\[
C(N + 1) - C(N) = C_1 m_1(n_1) + C_2 [m_2(N + 1) - m_1(n_1)] + C_3 [a - m_2(N + 1)] + C_4(N + 1) \\
- C_1 m_1(n_1) - C_2 [m_2(N) - m_1(n_1)] - C_3 [a - m_2(N)] - C_4(N) \\
\Rightarrow -(C_3 - C_2) [m_2(N + 1) - m_2(N)] + C_4,
\]

Where,

\[
m_2(N + 1) - m_2(N) = a \left[ 1 - (1-b_1)^n_1 (1-b_2)^{N+n_1} \right] - a \left[ 1 - (1-b_1)^n_1 (1-b_2)^{N-n_1} \right] \\
= a(1-b_1)^n_1 (1-b_2)^{N+n_1} b_2.
\]

Therefore,

\[
C(N + 1) - C(N) = -(C_3 - C_2) a(1-b_1)^n_1 (1-b_2)^{N+n_1} b_2 + C_4.
\]

When \( N = n_1 \)

\[
C(N + 1) - C(N) = \\
= -(C_3 - C_2) a(1-b_1)^n_1 b_2 + C_4
\]

**Case 1:** If \( ab_2(C_3 - C_2)(1-b_1)^n_1 > C_4 \).

Then there exists \( N_0 \) such that,
Subcase 1.

\[ ab_2(1-b_1)^n(1-b_2)^{N_0-n} = C_4 \] and \[ ab_2(1-b_1)^n(1-b_2)^{N_0-1-n} > C_4. \]

Then. \( N = N_0. \)

Subcase 2.

If \[ ab_2(1-b_1)'(1-b_2)^{N_0+n} > C_4 \] and \[ ab_2(1-b_1)'(1-b_2)^{N_0-n} < C_4. \]

\( N = N_0 \) Or \( N_0 - 1 \) depending on whether \( C(N_0 - 1) > C(N_0) \) Or not.

Case 2.

If \[ ab_2(C_3 - C_2)(1-b_1)^n \leq C_4 \] then \( N = n_1. \)
We further extend this problem with reliability constraint
\[ R(x/N) = e^{-(m_1(N+x)-m_2(N))}, \]
\[ \ln R(x/N) = m_2(N) - m_2(N + x), \]
\[ \ln R(x/N+1) - \ln R(x/N) = m_2(N+1) - m_2(N+1+x) - m_2(N) - m_2(N + x). \]

Solving the above we get,
\[ a(1-b_2)^n(1-b_2)^{N-n} \left[ b_1 - b_2 (1-b_2)^x \right] > 0. \]

So, \( R(x/N) \) is increasing.

**Case1.** If \( R(x/n_1) \geq R_o \),
Then \( N_2 = n_1 \).

**Case2.**
If \( R(x/n_1) < R_o \) then there exists \( N_1 \) and \( N_1 - 1 \) s.t.
\[ R(x/N_1 - 1) < R_o \& R(x/N_1) \geq R_o \]
Then \( N_2 = N_1 \)
\[ N^* = \text{Max}(N_0, N_2). \]

Graphically,

![Software Reliability v/s No of Test Cases](image)

The results for optimal release policy can be presented as:

\[ \text{Min} \quad C(N) \]
Subject to \( R(x/N) \geq R_o \)

Given that \( C_1 < C_2 < C_3 \)

1. If \( ab_2(C_3 - C_2)(1-b_2)^n \leq C_4 \) and \( R(x/n_1) \geq R_o \), then \( N^* = n_1 \).
2. If \( ab_2(C_3 - C_2)(1-b_2)^n \leq C_4 \) and \( R(x/n_1) < R_o \), then \( N_2 = N_1 \).
3. If \( ab_2(C_3 - C_2)(1-b_1)^{n} > C_4 \) and \( R(x/n_1) \geq R_o \), then \( N_* = n_i \).

4. If \( ab_2(C_3 - C_2)(1-b_1)^{n} > C_4 \) and \( R(x/n_1) < R_o \), then \( N^* = \text{Max}(N_0, N_1) \).

6. ILLUSTRATION

Illustration 1.

Now we derive optimal release policy with change point for the growth model described in section 5. Let the cost coefficients be given as \( C_1 = 8, \ C_2 = 12, \ C_3 = 18, \ C_4 = 15 \) and \( x=1 \). The parameter estimates are calculated using DS-2.

Here we see that

\[
N_0 = \frac{\ln\left(\frac{C_4}{ab_2}\right) - n_i \ln\left(\frac{1-b_1}{1-b_2}\right)}{\ln(1-b_2)} \quad \text{and} \quad N_1 = n_i + \frac{\ln(R_0)}{(1-b_2)^{x-1}a(1-b_1)} \frac{\ln(1-b_2)}{\ln(1-b_2)}. \]

We assume \( n_i = 6 \).

For the numerical under consideration \( N_0 = 152, \ N_2 = 100 \) and \( a b_2 (c_3 - c_2) > c_4 \) And \( R(x/n_1) < R_o \), therefore \( N^* = \text{max}(N_0, N_1) \) i.e. \( N^* = 152 \) \( C \left( N^* \right) = 9173 \).

7. CONCLUSION

In this paper we develop a generalized framework to develop various discrete SRGMs with change point using probability generating function. We further develop a release policy which minimizes the total cost function with a requisite reliability and find out the optimal no of test cases after which the software should be released in the market. The proposed models are validated on a data set cited in literature. Later in the paper we develop a cost function and minimize it subject to a reliability constraint and hence form a discrete software release time problem. Results of the software release time problem are illustrated using a numerical example.

REFERENCES


