Two Dimensional Flexible Software Reliability Growth Model And Related Release Policy

P. K. Kapur*1, R. B. Garg*2, Anu. G. Aggarwal*3 and Abhishek Tandon*4

1Department of Operational Research, University of Delhi, Delhi-110007
2N.I.E.T., Knowledge Park, Greater Noida, U.P., 3anuagg17@yahoo.com, 4abhishek.tandon@rocketmail.com

ABSTRACT
The prospective disastrous failure of software and the consequential damage is a scenario that is of great concern in terms of scientific disaster preparedness. Proper vigilance for such an event to enable mitigation of adverse impacts to the greatest degree possible requires modeling of software reliability to capture the risk of software failure. Traditionally, one-dimensional models have been used to evaluate the reliability and quality of the software, but such models are limited in their ability to capture the combined effect of testing time and testing coverage in removing the fault content in the software. In this paper we develop a two dimensional software reliability growth model which consists of testing time and testing coverage. We further develop a cost model with two dimensional mean value function which minimizes the cost and finds the optimal testing time and testing coverage with a predetermined level of reliability. A numerical example for the optimization problem is given and model is validated on two data sets.

KEYWORDS
Software Reliability, Two dimensional, Non-Homogeneous Poisson Process, Testing Coverage (TC), Fault detection (FD), Fault Correction (FC), Fault Detection Rate (FDC), Fault Removal Rate (FRR), Release Time.

1. INTRODUCTION
The propagation of software-based systems into the routine life of mankind in the last two decades has been beyond thoughts. Systems simple and complex are being rapidly computerized or updated with latest computing technology. It is surprising, because software that drives a computer system is a produce of human work and is very likely to contain faults. The last 30 years have seen the formulation of a number of software reliability growth models to predict the reliability and error content of software systems. These models are concerned with forecasting future system operability from the failure data collected during the testing phase of a software product. A plethora of reliability models have appeared in the literature. Software reliability is important because of our dependency on computer software system in our daily life and to the fact that software system cannot be made error free. In the last two decades different methodologies and techniques have been developed and put in to practice in the hope of producing high quality, low cost software systems. Generally the software development process is composed of four phases: requirement phase, design phase, coding and testing. The testing phase aims to detect and remove the latent software errors in order to ensure, as far as possible, error free operation of software in a given time. In other words, the testing phase quantifies the quality of the software in terms of its reliability. Thus software reliability is dependent on the number of errors remaining in the software.

Goel and Okumoto [2] proposed an SRGM, which describes the fault detection rate, as a non homogeneous poisson process (NHPP) assuming the hazard rate is proportional to remaining fault number. Later researchers proposed many SRGMs, which describe FD/or FC process by NHPP following the basic assumptions of GO model. Yamada et al. [12] proposed a modified exponential SRGM assuming the software contains two types of faults. The model is based on the observation that in the early stages of the software phase, the testing team removes a large number of simple faults (faults that are easy to remove) while the hard faults are removed in the later stages of the testing phase. Accordingly, they assumed the fault removal process to be the superposition of two NHPP, the first NHPP models the removal of the simple fault while the second models the removal of the hard faults. The FRR per remaining faults for the simple types is assumed to be greater than that of the hard type. The total FRR at the begging of the testing phase is equal to the FRR of the simple faults while it is equal to the FRR of the hard faults in the later stages of the testing phase. In the models of Ohba [4] and Yamada et al.[12], the assumption of constant FRR per remaining fault still holds for each fault type of faults. The total FRR per remaining fault is, however, a function of time. The first type is modeled by an Exponential model of Goel and Okamoto [2]. The second type is modeled by Delayed S-shaped model of Yamada et al. [12].

Testing Coverage is actually a structural testing technique in which the software performance is judged with respect to specification of the source code and the extent or the degree to which software is executed by the test cases [3, 7]. TC can help software developers to evaluate the quality of the tested software and determine how much additional effort is needed to improve the reliability of the software besides providing customers with a quantitative confidence criterion while planning to use a software product. Hence, safety critical system has a high coverage objective. The basic testing coverage measures are [3, 7, and 13]:
1. Statement Coverage: Decision
2. Condition Coverage:
3. Path Coverage:
4. Function Coverage:

Traditionally, one-dimensional models have been proposed in the literature with respect to testing time or testing coverage, although not much has been done to capture the collective effect of the testing coverage and the testing time. Ishii and Dohi [8] proposed a two dimensional software reliability growth model and their application. In this paper we develop a two-dimensional model which shows the combined effect of testing time and testing coverage to remove the faults lying dormant in the software. We assume that the number of faults removed in the software by a fixed time is dependent on the total testing resources available to the testing team. This testing resource is a fusion of both testing time and testing coverage. We employ Cobb-Douglas production function [8] to demonstrate the effect of both testing time and testing coverage in removing the faults in the software. Inoue proposed a two dimensional software reliability growth model with testing coverage using the Cobb Douglas production function. Further in this paper we develop a release policy which gives us an optimal value of the testing time and testing coverage which minimizes the total testing cost subject to a pre requisite level of reliability.

2. TIME DEPENDENT MODEL
The time dependent behavior of fault removal process is explained by a Software Reliability Growth Model (SRGM). In literature, several SRGMs have been proposed to measure the reliability during the testing phase. Most of these can be categorized under Non Homogeneous Poisson Process (NHPP) models. The assumption that governs these models is, ‘the software failure occurs at random times during testing caused by faults lying dormant in software.’ And, for modeling the software fault detection phenomenon, counting process \( \{ N(t) ; t \geq 0 \} \) is defined which represents the cumulative number of software faults detected by testing time \( t \). The SRGM based on NHPP is formulated as:

\[
Pr \{ N(t) = n \} = \frac{m(t) \cdot \exp(-m(t))}{n!}, \quad n = 0,1,2,...
\]

Where \( m(t) \) is the mean value function of the counting process \( N(t) \).

2.1 Testing Coverage Based Modeling Notations
\( m(t) \): Expected number of faults identified in the time interval \((0,t]\).
\( c(t) \): Testing coverage as a function of time \( t \)
\( v \): Constant
\( N \): Constant, representing the number of faults lying dormant in the software at the beginning of testing.

The testing coverage based software reliability growth model can be formulated as follows:

\[
\frac{dm(t)}{dt} = \frac{c'(t)}{1-c(t)}(N-m(t))
\]

Here \( c'(t) \) defines the percentage of the coded statements that has been observed till time \( t \). So, \( 1 - c(t) \) defines the percentage of the coded statements which has not yet been covered till time \( t \). Then, the first order derivative of \( c(t) \), Denoted by \( c'(t) \), represents the testing coverage rate.

Therefore, function \( \frac{c'(t)}{1-c(t)} \) can be taken as a measure of the fault detection rate [7]. In one dimensional SRGM with testing coverage we need to define coverage function \( c(t) \) although in a two dimensional modeling approach we need not define a coverage function and it can be estimated directly from the data.

3. Kapur-Garg Model
In 1992 Kapur and Garg [6] developed an S-shaped model with an assumption that when we remove the faults in the software some additional faults in the software are removed without actually affecting the system. This model was based on the assumption of Non-Homogeneous Poisson Process. The basic assumptions of the model are as follows:
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 time moment.

**Notations**

\( a \) : Rate at which a fault is detected/removed in the software
\( b \) : Rate at which an additional fault is detected/removed in the software
\( m_f \) : Mean Number of failures corresponding to testing time \( t \)
\( m_r \) : Mean Number of faults detected corresponding to testing time \( t \)

The differential equation of the representing the rate of change of cumulative number of faults detected in time \( t \) is given as

\[
m_r'(t) = a(N-m_r(t)) + \frac{b(N-m_r(t))}{N}
\]

The mean value function of the number of faults detected in time \( t \) is given as:

\[
m_r(t) = \frac{Na\left(\exp(a+b)t - 1\right)}{b+a\exp((a+b)t)}
\]
4. TWO –DIMENSIONAL MODELING

To cater the combined effect of testing time and testing coverage we use Cobb-Douglas production function. In economics, the Cobb–Douglas functional form of production functions is widely used to represent the relationship of an output to inputs. It was proposed by Knut Wicksell (1851–1926), and tested against statistical evidence by Charles Cobb and Paul Douglas in 1900–1928.

\[ \tau \cong s^\alpha u^{1-\alpha} \quad 0 \leq \alpha \leq 1 \]

Where
- \( \tau \): testing resources
- \( s \): testing time
- \( u \): testing coverage
- \( \alpha \): Effect of testing time

Let \( \{N(s,u), s \geq 0, u \geq 0\} \) be a two-dimensional stochastic process representing the cumulative number of software failures by time \( s \) and testing coverage \( u \). A two-dimensional NHPP with a mean value function \( m(s,u) \) is formulated as:

\[ \Pr(N(s,u) = n) = \frac{m(s,u)^n}{n!} \exp(-m(s,u)), \quad n = 0, 1, 2... \]

and

\[ m(s,u) = \int_0^u \int_0^s \lambda(\zeta, \xi) d\zeta d\xi \]

4.1 Two –Dimensional S-Shaped Model

In this paper we develop a two dimension S-shaped model determining the combined effect of testing time and testing coverage. We define some additional notations as follows:

- \( m_r(s,u) \): Mean Number of faults detected corresponding to Coverage \( u \) and time \( s \)
- \( f_s(s,u) \): Mean Number of failures corresponding to Coverage \( u \) and time \( s \)

The differential equation of the representing the rate of change of cumulative number of faults detected w.r.t. to the total testing resources is given as:

\[ R(\omega|s_s, u_s) = \exp\left[-m(s_s + \omega / \Xi) - m(s_s, u_s / \Xi) \right] \]

Where \( \Xi \) indicates the set of parameter estimates of a two-dimension SRGM.

**Data set 1 (DS-1):** Coverage Data set with 796 cumulative numbers of faults [13].

**Data set 2 (DS-2):** Coverage Data set with 1196 cumulative number of faults [13].
6. ESTIMATION OF PARAMETERS MODEL VALIDATION AND COMPARISON CRITERIA

6.1. Model Validation

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

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.

6.2.1. 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”?

6.2.2. The Mean Square -Error (MSE):

The model under comparison is used to simulate the fault data, the difference between the expected values, \( \hat{m}(t_i) \) and the observed data \( y_i \) is measured by MSE as follows.

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

where \( k \) is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit [6].

6.2.3 Bias

The difference between the observation and prediction of number of failures at any instant of time \( i \) is known as PEi (prediction error). The average of PEs is known as bias. Lower the value of Bias better is the goodness of fit [6].

<table>
<thead>
<tr>
<th>Table 1: Model Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>DS-1 MSE</td>
</tr>
<tr>
<td>Bias</td>
</tr>
<tr>
<td>Adj ( R^2 )</td>
</tr>
<tr>
<td>Variation</td>
</tr>
<tr>
<td>RMPSE</td>
</tr>
<tr>
<td>DS-2 MSE</td>
</tr>
<tr>
<td>Bias</td>
</tr>
<tr>
<td>Adj ( R^2 )</td>
</tr>
<tr>
<td>Variation</td>
</tr>
<tr>
<td>RMPSE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2: Parameter Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Two dimensional DS-1</td>
</tr>
<tr>
<td>One dimensional GO</td>
</tr>
<tr>
<td>One dimensional KG</td>
</tr>
<tr>
<td>Two dimensional DS-2</td>
</tr>
<tr>
<td>One dimensional GO</td>
</tr>
<tr>
<td>One dimensional KG</td>
</tr>
</tbody>
</table>

The mean square error of our proposed two dimensional models is better than the existing one dimensional software reliability growth model.
7. OPTIMAL RELEASE PROBLEM

A problem of great importance in a software development project is when to stop testing and release it for operation as it affects both the software reliability and the total cost of the project. The longer the testing phase is, the higher is the reliability and the smaller is 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 length of testing phase calls for trade-offs between these two conflicting facets. Such a problem is known as “Optimal Software Release time Problem”. A lot of research has been done in the area of release time optimization during last three decades and several new cost models have been developed in the literature. Now we define some additional parameters associated with the release policy under discussion:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s^*$</td>
<td>Optimal time to stop testing or release time</td>
</tr>
<tr>
<td>$u^*$</td>
<td>Optimal testing coverage before the release of the software</td>
</tr>
<tr>
<td>$C_1$</td>
<td>Cost of detecting one fault causing failure during testing phase</td>
</tr>
<tr>
<td>$C_2$</td>
<td>Cost of detecting an additional fault during testing phase</td>
</tr>
<tr>
<td>$C_3$</td>
<td>Cost of detecting one fault causing failure during operational phase</td>
</tr>
<tr>
<td>$C_4$</td>
<td>Cost of detecting an additional fault during operational phase</td>
</tr>
<tr>
<td>$C_5$</td>
<td>Testing cost per unit time</td>
</tr>
<tr>
<td>$R_O$</td>
<td>Aspiration level for the software reliability</td>
</tr>
</tbody>
</table>

We define following mean value functions for the release policy:

$$m_f(u) = Na b \ln \left[ \frac{a + b}{a b^{(a+b)s^\alpha} + b} \right]$$

$$m_r(u) = Na \ln \left[ \frac{a + b}{ae^{a+b} + b} \right]$$

Where $r = (a + b)s^\alpha$

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

$$C(u) = Cm_f(u) + C[m_f(u) - m_f(u)] + C[m_r(u) - m_r(u) - m_r(u)] + C_5 u^\alpha$$

$$C(u) = 0$$

$$\Rightarrow$$

$$m_f(u) + (D-1)m_f(u) = \frac{C_5(1-\alpha)\left(\frac{s}{\alpha}\right)\alpha}{C_4 - C_2}$$

where $D = \frac{C_5}{C_4 - C_2}$

Or,

$$\frac{Na(a + b)^2e^{(a+b)s^\alpha} + \alpha}{\left(b + ae^{(a+b)s^\alpha} + \alpha\right)} +$$

$$(D-1)Na(a + b) = \frac{C_5}{b + e^{(a+b)s^\alpha}}$$

$$(D-1)Na(a + b) = \frac{C_5}{C_4 - C_2}$$

When $u = 0$ L.H.S $= NaD$
at \( u = \infty \), LHS = 0

if \( NaD > \frac{C_5}{C_4 - C_2} \)

\[ m'(u) + (D-1)m'(u) = \frac{Na(a+b)^2 e^{(a+b)u}}{(b + ad(a+b)e^{a+b}u^{\alpha - 1})} \]

\( m'(u) + (D-1)m'(u) = \frac{Na(a+b)^2 e^{(a+b)u}}{b + ad(a+b)e^{a+b}u^{\alpha - 1}} + (D-1) \frac{Na(a+b)}{b + ad(a+b)e^{a+b}u^{\alpha - 1}} \)

Its nature depends upon

\[ \frac{(a+b)e^{(a+b)u^{\alpha - 1}}}{(b + ad(a+b)e^{a+b}u^{\alpha - 1})} + \frac{(D-1)Na(a+b)}{b + ad(a+b)e^{a+b}u^{\alpha - 1}} \]

Partially differentiating w.r.t. \( u \), we get

\[ \left( b - ae^{(a+b)u^{\alpha - 1}} \right)(a+b) - (D-1)a \left[ b + ae^{(a+b)u^{\alpha - 1}} \right] \]

(4)

When \( u = 0 \), we get

\[ (a+b)(b - ad D) \]

And \( X \) is decreasing

So if \( aD \geq b \) and \( NaD > \frac{C_5}{C_4 - C_2} \)

There exists a finite \( u = u_0 \) which satisfies

(1)

If \( NaD \leq \frac{C_5}{C_4 - C_2} \) & \( aD \geq b \) then \( u = 0 \) minimizes \( C(u) \).

If \( aD < b \), then though (4) is decreasing it will be positive and then negative. It will be zero when

\[ b - ae^{(a+b)u^{\alpha - 1}} = (a+b) \]

\[ (D-1)a \left[ b + ae^{(a+b)u^{\alpha - 1}} \right] = X \]

Let

\[ (D-1)a \left[ b + ae^{(a+b)u^{\alpha - 1}} \right] = X \]

When \( u = 0 \), we get

\[ (a+b)(b - ad D) \]

And \( X \) is decreasing

So if \( aD \geq b \) and \( NaD > \frac{C_5}{C_4 - C_2} \)

Now (4) will be positive for \( u < \), negative for \( u > u_m \) and zero at \( u = u_m \)

If \( aD < b \) & (1) \( \leq \frac{C_5}{C_4 - C_2} \), then for \( u = u_m \), \( \frac{dC(u)}{du} > 0 \) for \( u > 0 \) or if

\[ aD < b \) & \( NaD \geq \frac{C_5}{C_4 - C_2} \)

finite and unique

\[ u = u_2 (u_m) \] satisfying (3) exists.

\[ aD < b \) & \( NaD \geq \frac{C_5}{C_4 - C_2} \)

\( u = u_2 (u_m) \)

Satisfying (3) exists, Moreover if \( C(0) < C(u_2) \) then there exists a finite and unique \( u = u_c (u_m) \) satisfying \( C(u_c) = C(u_2) \).

If \( C(0) > C(u_2) \) then a finite and unique \( u = u_c \) & \( u = u_f \) exists satisfying \( C(0) = C(u_c) = C(u_f) \).

The expected software reliability \( R(x/s) = e^{-m_f s} \) is defined as the probability that the software does not fail in the time interval \((s, s + y)\) and a fixed testing coverage \( u \), given that the last failure occurred at \( s, s \geq 0 \) \((y > 0)\). Here \( R(y/s) \), software reliability function at time \( s \) and testing coverage \( u \) is given by

\[ R(x|u) = e^{-(m_f y u - m_f s)} \]

\[ \ln R(x|u) = m_f (s,u) - m_f (s+y,u) \]

\[ \frac{\partial \ln R(x|u)}{\partial u} = m'_f (s,u) - m'_f (s+x,u) > 0 \]

\[ R(x/0) = e^{-(m_f s)} \]

\[ R(x/\infty) = 1 \]

\[ R(x/0) < R_o \]

So finite and unique \( u_1 > 0 \) exists where \( R(x/u_1) = R_o \).

Extending the optimization problem to a reliability constraint, we get:

Min \( C(s,u) \)

Subject to \( R(s/x(u)) \geq R_o \).

Assume \( C_1 > C_2 \), \( C_1 > C_2 \), \( C_1 > C_2 \), \( C_1 > C_2 \), \( 0 < R_o < 1 \), \( y \geq 0 \).

We first assume \( s \) is known from cost and find \( u \), using this \( u \) now from reliability we find \( s \) and repeat this process till \( s \) and \( u \) stabilize. The results for optimal release policy can be presented as:

Theorem.1

When \( aD \geq b \)

\[ (i) \ NaD > \frac{C_5}{C_4 - C_2} \]
Using data set 1. We discuss the value of \( s \) higher than the assumed value of \( s \), therefore the value of \( s^* = 9 \) and \( u^* = 35 \).

**Theorem 2**

When \( aD < b \)

(i) \( NaD \geq \frac{C_3}{C_4 - C_2} \) \( u^* = \max(u_2, u_1) \) for \( R(x/0) < R_o < 1 \)

(ii) \( NaD < \frac{C_3}{C_4 - C_2} \) and (3) \( \geq \frac{C_5}{C_4 - C_2} \) for \( u = u_m \)

(a) \( C(0) = C(u_x) \)

(b) \( u^* = \max(u_2, u_1) \) for \( R(x/0) < R_o < 1 \)

(c) \( u^* = 0 \) or \( u_x \) for \( 0 < R_o < R(x/0) (b)C(0) < C(u_x) \)

(d) \( u^* = 0 \) or \( u_x \) for \( u = u_m \)

(e) \( C(0) > C(u_x) \)

(ii) \( u^* = u_1 \) for \( u_1 < u_c \) (iii) \( u^* = u_c \) or \( u_x \) for \( u_1 = u_c \) or

(iv) \( u^* = \max(u_2, u_1) \) for \( u_1 > u_c \)

8. **Numerical Example**

Using data set 1 \( N = 9, a = .022, b = .0001 \). We discuss the optimal release policy for software system. We assume \( C = 1, C = 15.9, C = 25, C = 21, C = 45, s = 9 \)

\[ x = .30, \text{ and } R = .9 \]

\[ \text{Since } N \text{ad} > \frac{C}{C - C} \]

\( u \) is estimated as 22. Moreover \( R(x/0) < .8 \)

\( u \) is estimated as 35. Using theorem 2 (i)

\( u^* = \max(u_2, u_1) \) for \( R(x/0) < R_o < 1 \) \( u^* = 22, 35 \)

Using the value of \( u^* \) in the reliability function we get the value of \( s \) higher than the assumed value of \( s \), therefore the optimum value of \( s \) using assumed \( u^* \) is 9. Therefore the value of \( s^* = 9 \) and \( u^* = 35 \).

9. **Conclusion**

In this paper we have used a two dimensional approach to develop a flexible software reliability growth model using Cobb Douglas production function so as to capture the combined effect of testing time and testing coverage. Later in the paper, we have developed a cost model incorporating the concept of two dimensional modeling for deriving the optimal release time and optimal coverage before the release. It involves minimization of expected software cost subject to a desired level of reliability. The models obtained are validated on real data sets & numerical illustration is given for the optimal release time problem.

10. **References**


