Chapter-5

Some Related problems in Software Reliability

In this chapter we have discussed important management problems related to software. We have made use of the concept of features addition done in the software regularly. The up-gradation model has been characterized by increasing the number of features in the software that gives the firm competitive edge in the market. On the other hand, continuous up-gradation of software’s also brings complexity in the systems once it failed to work properly. Many software reliability growth models (SRGM) have been proposed over past three decades that estimate the reliability of a software system as it undergoes changes through the removal of failure causing faults. But unfortunately most of the models didn’t consider anything about the increase in failure rate once an up-gradation is made on the software. The objective of this part of the chapter is to propose the software reliability growth model that incorporates the effect of enhancement of features on software during testing and debugging process. Results have been supplemented with numerical examples. In the later part of the chapter we have directed our work from working in single dimensions to multi-dimensional framework. The new outlines of competition and collaboration that have arisen in software engineering as a result of the globalization process have an impact on the entire software process. Therefore to capture the combined effect of testing time and testing coverage we have proposed a two dimensional software reliability growth model. We have made use of the famous cobb-douglas production function to develop the two dimensional model. We have presented two software reliability growth models incorporating logistic distribution function and Exponentiated Exponential distribution function respectively. The proposed models are validated on real data set.

This chapter is based on the following papers:


5.1 A Software Reliability Growth Model embodying features intensification

The software industry can be considered as the archetypal high technology industry where innovation and knowledge creation form the primary fuel for continued firm growth. Often the rate of innovation achieved by a firm shapes its evolutionary path as well as its future growth. The innovation-related factors like development process, software testing and debugging process and team structure have significant impact on a firm’s future growth potential. In the last few decades it has been observed that the world of software development management (i.e. new product development, technology alliance etc.) has evolved rapidly due to the intensified market competition. In particular the use of continuous up-gradation model of software products fast becoming commonplace due to the shrinking budget, expanding system requirements and on the other hand accelerating rate of software enhancement. The up-gradation of the system is done by extending it through add-ons, interfacing with other applications etc. The growing trend towards up-gradation of software’s has taken the original concept of reprocess it into a completely different arena and due to that it has also presented many challenges to software developers attempting to enter this new arena. Due to the continuous up-gradation of softwares, they are rapidly becoming an integral part of nearly every engineered product, controlling the manufacturing process for products, commercial aircrafts, nuclear power plants, medical devices, weapon systems, aerospace systems, automobiles, public transportation systems, and so on. At the same time it brings complexity in the systems once it fails to work properly. Software failures may occur due to errors, indistinctness, oversights or misspecification that the software is supposed to satisfy, incompetence in writing code, inadequate testing, incorrect or unexpected usage of diction of software failures can contribute substantially to long-term financial success and are an effective strategy to increase the reliability of the software. Software Reliability is an important attribute of software quality. It is hard to achieve as the complexity associated with software tends to be high. Pan (1999) argued that due to the rapid growth of software size and ease of doing so by upgrading the software, a high degree of complexity is brought in it and due to that it becomes hard to reach a certain level of reliability. Mathematical models have
been proved to be useful tools for understanding the structure and functioning of software, predicting the reliability of software and prescribing the best course of actions under known constraints. In recent years, there has been a growing interest to predict the link between the rates of failure and the reliability of software. Software reliability is the probability that given software will be functioning correctly under a given environment during a specified period of time. Many software reliability growth models (SRGM) have been proposed over the past three decades that estimate the reliability of a software system as it undergoes changes through the removal of failure causing faults. The Goel-Okumoto model (1979) is among the most quoted publications in this area. The model is a purely exponential and based on the assumption that the faults are uniformly distributed where each fault has an equal chance of detection. Yamada et al (1983) and Kapur et al (1995) proposed a software reliability growth model assuming that each fault detection leads to exactly one removal. Kapur and Garg (1992), Obha (1984) and Bittani et al. (1988) proposed models where fault detection rate is defined as function of number of faults already removed. Many software reliability growth models were proposed in literature based on non homogenous poisson process (NHPP). Some of them are as follows; Obha and Yamada (1984); Yamada, Ohtera and Narihisa (1986); Kapur, Garg and Kumar (1999); Kenny (1993); Yamada, Tamura and Kimura (2003) etc. They were based on the assumption that the initial fault contents of software remain the same for entire life cycle. But unfortunately, these models do not consider anything about the drastic increase in failure rate software may experience each time an upgrade is made. One of the objectives of this part of the chapter is to understand the role of enhancement of features during test-phase that aid to increase in failure contents of software.

5.1.1 Contribution of Study

Significant improvement in software reliability calls for innovative methods for developing software, determining its readiness for release, and predicting field performance. Software products which are introduced in the market can be one-off type (i.e. no enhancements is made in the software) or can be a product which is periodically upgraded with new features upto a level. To acquire the ability to produce and market new products, companies make huge investments and the failure of such products can
be damaging. Therefore to optimize the companies’ goals, they try to reduce the risk by staggering the new ideas to a sequence of product-features introduced over a period of time and satisfying multiple market segments. Since the pioneering work of Goel-Okumoto model (1979), many authors have proposed software reliability growth models to measure the failure rate of software.

This high level of interest in measuring the failure rate has resulted in a large body of publications like Musa, Iannino and Okumoto (1987); Yamada and Osaki (1983); Abdel, Chen and Littlewood (1986); Kapur and Garg (1999) etc. All the models are based on the assumption that when the software is first manufactured, the initial number of faults is high but then decreases as the fault components are identified and removed or the components stabilize. The software then enters the useful life phase where more faults are removed and the failure rate levels of gradually. The typical software failure curve experienced by traditional software reliability growth model can be depicted by the figure 5.1. Thus, the traditional software reliability growth model failed to capture the error growth due to the software enhancements in the test phase. In the useful-life phase as the software firm introduces new add-ons or features on the basis of the user needs, software will experience a drastic increase in failure rate each time an upgrade is made. The failure rate levels off gradually, partly because of the defects found and fixed after the upgrades. Figure 5.2 depicts the increase in failure rate due to the addition of new features in the software. Due to the feature upgrades, the complexity of software is likely to be increased as the functionality of software is enhanced. Even fixing bugs may induce more software failures by fetching other defects into software. But if the goal of the firm is to upgrade the software by enhancing its reliability, then it is possible to incur a drop in software failure rate that can be done by redesigning or re-implementing some modules using better engineering approaches (Pan 1999). The objective of this section of chapter is to propose a software reliability growth model under the assumption that software will experience a drastic increase in failure content each time an upgrade is made by the software firm.
Software Reliability Growth Models (SRGMs) play an important role due to their ability to predict the fault detection / removal phenomenon during testing. Several classes of SRGMs have been proposed and validated on test data in the literature. A proliferation of software reliability models has emerged as people try to understand the characteristics of how and why software fails, and try to quantify software reliability. Over 200 models have been developed since the early 1970s, but how to quantify software reliability still remains largely unsolved. One group of models that has been
widely used and researched is the Non-homogeneous Poisson Process (NHPP) models. The proposed model is also based on NHPP framework. Using the logistic rate function, the K-G model (1992) (which is the building block of this paper) can be described by following mathematical structure:

$$\frac{dm(t)}{dt} = \frac{b}{1 + \beta e^{-bt}} (a - m(t))$$  \hspace{1cm} (5.1)

Where $m(t)$ is the cumulative number of faults removed in the software by time $t$; $a$ is the finite number of fault content present in the software $b$ is the constant fault detection rate and $\beta$ is the learning parameter.

### 5.1.2 Notations

- $a$: Expected number of faults in the software
- $b_1$: Detection rate before the time $s$.
- $b_2$: Detection rate after the time $s$.
- $m(t)$: Expected Number of faults removed.
- $S$: Time after which we start adding new features.
- $\alpha$: Rate of fault addition due to adding a new feature in the software.
- $\beta_1$: learning parameter before the time $s$.
- $\beta_2$: learning parameter after the time $s$.

### 5.1.3 Basic Assumptions

Let $\{N(t); t \geq 0\}$ be a counting process representing the cumulative number of software failures by time $t$. The $N(t)$ process is shown to be a NHPP with a mean value function $m(t)$. Mean value function represents the expected number of faults removed by time $t$.

$$\Pr\{N(t) = n\} = \left(\frac{m(t)}{n!}\right)^n e^{-m(t)}, \hspace{1cm} n = 0, 1, 2, \ldots$$  \hspace{1cm} (5.2)

And

$$m(t) = \int_0^t \lambda(x)dx$$  \hspace{1cm} (5.3)
Chapter 5

Some Related problems in Software Reliability

The proposed model is based upon the following basic 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 time moment.
5. Up-gradation is done continuously after a fix time s.

5.1.4 Model Development

In this section, we formulate probability distribution based software reliability growth models incorporating the affect of adding new features in the software system. As discussed, integrating new features increases complexity in the software and can be the cause of more faults in the system. Even fixing bugs may induce more software faults by fetching other defects into software. In general, before adding any additional features in the system (i.e. before time s) the initial fault contents in the software system remain constant and are detected and eliminated during the testing phase, and the number of faults remaining in the software system gradually decrease as the testing process goes on. On the other hand, adding new features in the software introduces more faults in the system. Assuming the software firm starts incorporating additional features after time ‘s’ and each additional feature puts in faults at the rate ‘α’, the proposed model shows continuous flow dynamic character. The model for different situations can be described as follows:

\[
\frac{dm(t)}{dt} = \begin{cases} 
\frac{b_1}{1 + \beta_1 e^{-bt}}(a - m(t)) & 0 \leq t < s \\
\frac{b_2}{1 + \beta_2 e^{-bt}}(a(1 + \alpha t) - m(t)) & t > s 
\end{cases}
\]  

(5.4)

Under the initial condition at
On solving the equation (5.4), the mean value function is obtained as follows:

\[
m(t) = \begin{cases} 
  a \left( \frac{1-e^{-h \cdot t}}{1+\beta_e e^{-h \cdot t}} \right) & \text{if } t \leq \tau \\
  a \left[ \frac{1-e^{-h \cdot t}}{b} \left( \frac{1-e^{-b \cdot (t-\tau)}}{1+\beta_e e^{-b \cdot (t-\tau)}} + \frac{1+\beta_e e^{-b \cdot (t-\tau)}}{e^{-b \cdot (t-\tau)}-e^{-b \cdot (t-\tau)}} \right) \right] & \text{if } t > \tau
\end{cases}
\]

(5.5)

From Equation (5.5) it can be observed that the parameter \( \alpha \) (i.e. rate of fault addition due to additional feature added) plays a significant role during estimation of the expected number of faults removal.

### 5.1.5 Data Analyses

The parameters of the models have been estimated using the statistical package SPSS. To illustrate the estimation procedure and application of the SRGM (existing as well as proposed) we have validated the proposed model on two different data-sets, which are as follows:

**Data set 1 (DS-1)**

The first data set (DS-1) was collected during 35 months of testing a radar system of size 124 KLOC where 1301 faults were detected during the testing. This data is cited from Brooks and Motley (1980). The change-point for this data set is 17th month.

**Data set 2 (DS-2)**

The second data set (DS-2) was collected during 19 weeks of testing a real time command and control system, where 328 faults were detected during the testing. This data is cited from Ohba (1984). The change-point for this data set is 6th week.
### Table 5.1: Parameter Estimates of the K-G Model and the proposed model

<table>
<thead>
<tr>
<th>Parameters</th>
<th>DS-1</th>
<th>DS-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K-G Model</td>
<td>Proposed Model</td>
</tr>
<tr>
<td></td>
<td>(eqn. 5.1)</td>
<td>(eqn. 5.5)</td>
</tr>
<tr>
<td>$a$</td>
<td>1356</td>
<td>1322</td>
</tr>
<tr>
<td>$b_1$</td>
<td>0.177</td>
<td>0.2133</td>
</tr>
<tr>
<td>$b_2$</td>
<td></td>
<td>0.2112</td>
</tr>
<tr>
<td>$\beta_1$</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>$\alpha$</td>
<td></td>
<td>0.063</td>
</tr>
</tbody>
</table>

### Table 5.2: Model Comparison

<table>
<thead>
<tr>
<th>Parameters</th>
<th>DS-1</th>
<th>DS-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K-G Model</td>
<td>Proposed Model</td>
</tr>
<tr>
<td></td>
<td>(eqn. 5.1)</td>
<td>(eqn. 5.5)</td>
</tr>
<tr>
<td>$R^2$</td>
<td>0.994</td>
<td>0.998</td>
</tr>
<tr>
<td>bias</td>
<td>-3.071</td>
<td>1.833</td>
</tr>
<tr>
<td>variation</td>
<td>619.341</td>
<td>76.494</td>
</tr>
<tr>
<td>$RMSPE$</td>
<td>620.07</td>
<td>76.516</td>
</tr>
<tr>
<td>$MSE$</td>
<td>41523</td>
<td>19906</td>
</tr>
</tbody>
</table>
The proposed model has been compared with the K-G model on the basis of different comparison criteria. These are: Mean Square Fitting Error (MSE), Coefficient of Multiple Determination ($R^2$), Bias, Variation and Root Mean Square Prediction Error. Table 5.2 summarizes the value of all the comparison criteria for each of the models for two different datasets. From Table 5.2, it is clear that MSE, Bias, Variation and RMSPE of the proposed model are the least in comparison to those of the K-G model. Also, the adjusted $R^2$ value of the proposed model is relatively higher in comparison to that of the K-G model for the two different datasets. From Table 5.2 and Figures 5.3 and 5.4, it can be concluded that the proposed model works better for the two different datasets. Apart from goodness of fit criteria, the parameters of the proposed model provide important information. From Table 5.1, it can be seen that the estimates of rate of error generation due to enhancement of features in the software $\alpha$ are relatively high, which suggests that up-gradation of the software plays a vital role in increasing the fault contents.
Figure 5.3: Actual faults vs estimated Faults for DS-I

Figure 5.4: Actual Faults vs Estimated Faults for DS-2
5.2 Some Flexible Software Reliability Growth Models Using Two-Dimensional Approach

A huge number of software reliability growth models (abbreviated as SRGMs) (Kapur et al 1999, 2012, Okumoto 1979, Yamada et al 1992, Musa et al 1987) which assess software reliability quantitatively have been proposed so far. However, almost all of the SRGMs have been developing under the assumption that the software reliability growth process depends only on the testing-time as the software reliability growth factor essentially. On the other hand, it is known that a software reliability growth process in a testing-phase is influenced by the following several software reliability factors: the test execution-time, the testing-skill, the testing coverage, and so forth. Yamada et al. (1986, 1993) proposed a testing-effort dependent SRGM based on a non homogeneous assumption that the fault-detection rate is proportional to the testing-effort expenditure. Fujiwara and Yamada (2001) developed a testing-domain dependent SRGM which incorporates the testing-skill of test-case designers, where the testing-domain means a set of testing-paths in the software system to be influenced by executed test-cases. Inoue and Yamada (2004) discussed a testing-coverage dependent SRGM by characterizing the relationship between the testing-coverage attainment process and the software reliability growth process mathematically and developing a testing-coverage function to describe a time-dependent behaviour of a testing-coverage attainment process with the testing-skill of test-case designer. However, such SRGMs, which consider with the effect of the software reliability growth factors to the software reliability growth processes, depend only on the testing-time essentially. But does the software reliability growth process depend only on the testing-time in fact? One of the answers to the problems mentioned above is developing an SRGM which depends on the testing-time and other reliability growth factors simultaneously. Developing such an SRGM would be more feasible for describing a software reliability growth process in an actual testing-phase. In recent years, Ishii and Dohi et al (2004) have proposed a two dimensional software reliability growth model and their application. They investigated the dependence of test-execution time as a testing effort on the software reliability assessment, and validate quantitatively the software reliability models with two-time scales. Inoue and Yamada (2008, 2009) also proposed two dimensional software reliability growth models. However their modeling framework was not a direct representative of using mean value functions to represent of fault removal process.
They discussed software reliability assessment method by using two dimensional Weibull type SRGM. This study aims to compare the predictive capability of two popular software reliability growth models (SRGM), say flexible logistic growth and exponentiated exponential growth. We present an exponentiated exponential growth model, which can capture the increasing or constant, decreasing nature of the failure occurrence rate per fault.

In this part of the chapter we discuss two dimensional SRGMs which enable us to expect more feasible software reliability assessment than the conventional software reliability measurement approach. To start with, we have defined one-dimensional unified approach for describing failure-occurrence or fault-detection phenomenon before discussing our two-dimensional software reliability growth modelling framework. After that, we have presented a unified framework for developing two-dimensional SGRM with respect to testing time and testing coverage.

Further, we conduct goodness-of-fit comparisons with several SRGMs proposed so far, and show numerical examples of software reliability assessment based on our model by using actual fault count data collected along with testing-coverage data.

### 5.2.1 Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>Number of faults lying dormant in the software at the beginning of testing.</td>
</tr>
<tr>
<td>$m(t)$</td>
<td>Expected number of faults removed in the time interval $(0, t]$.</td>
</tr>
<tr>
<td>$b$</td>
<td>Constant fault detection/isolation/correction rate.</td>
</tr>
<tr>
<td>$f(t)$</td>
<td>Probability density function</td>
</tr>
<tr>
<td>$F(t)$</td>
<td>Probability distribution function.</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Effect of testing parameter</td>
</tr>
<tr>
<td>$\phi$</td>
<td>Shape parameter for exponentiated exponential function</td>
</tr>
<tr>
<td>$\varphi$</td>
<td>Scale parameter for exponentiated exponential function</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Constant</td>
</tr>
</tbody>
</table>

### 5.2.2 Finite Failure Software Reliability Growth Modeling

The NHPP models are based on the assumption that the software system is subject to failures at random times caused by manifestation of the remaining faults in the system. Hence, NHPP are used to describe the failure phenomenon during the testing phase.
Apart from the usual assumptions that have been discussed earlier part of the chapter; the proposed models are based upon the following basic assumptions.

1. The failure observation and fault removal phenomenon is modeled using a 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 debugging effort takes place to find the cause of the failure to remove it.

Based on the above assumptions the differential equation describing the rate of change in the cumulative number of faults is given as follows:

\[
\frac{dm(t)}{dt} = \frac{f(t)}{1-F(t)}(a-m(t)) = h(t)(a-m(t))
\]  

(5.6)

where,

\( h(t) \) is defined as the hazard rate with which the faults are detected /corrected in the software system and \((a-m(t))\) denotes the expected number of faults remaining in the software at time \(t\).

Solving Equation (5.6) using the initial condition that at \(t=0, m(t)=0\), we get

\[
m(t) = a.F(t)
\]  

(5.7)

Equation (5.7) is the Generalised NHPP SRGM model. Substituting different types of distribution functions, i.e. different values for \(F(t)\) in (5.7), we can obtain different mean value functions corresponding to them (Pham 1993, Kapur et al 2012)

**5.2.3 Two Dimensional Modeling Framework**

It is a well accepted fact that modeling is a strong tool to plan the steps in development of a system. A model explains the system at different levels of abstraction. And in software reliability a most important tool that can evaluate the software quantitatively, develops test status, schedules status and monitors the changes in reliability performance is Software Reliability Growth Model (SRGM). During the last three decades, a large number of SRGMs have been proposed in literature. However, almost all of the SRGMs are developed under the assumption that software reliability growth
Some Related problems in Software Reliability

The process depends only on testing-time. Later some testing resource dependent SRGMs were also developed. Also, there exists testing coverage based SRGMs in the literature. But all these models do not take into account the simultaneous effect of time and resources or fails to consider the concurrent effect of time and coverage on cumulative number of faults removed from software. Such models can be termed as one-dimensional software reliability growth models. Lately, some Two-dimensional software reliability models have been developed to assess the software quantitatively (Inoue and Yamada (2008, 2009), Ishii and Dohi 2006). The need for developing a two dimensional model is an ideal solution to the problem of software reliability at the hands of software engineers. Two dimensional models are used to capture the joint effect of testing time and testing coverage on the number of faults removed in the software. The traditional one dimensional model has been dependent upon the testing time, testing effort or testing coverage. However, if the reliability of a software is measured on the basis on the number of hours spent on testing the software or the percentage of software that has been covered then the results are not conclusive. To cater the need of high precision software reliability we require a software reliability growth model which caters not only the testing time but also the testing coverage of the software i.e. the percentage of code covered of the software. For this we develop a two dimensional software reliability growth model incorporating the joint effect of testing time and testing effort on the number of faults removed in the software. The two dimensional model developed in this section is based on the Cobb Douglas production function (discussed in chapter 1).

Testing is oriented to 'detection'. The testing team has many resources of testing to make sure that software hence formed is of quality. These include software testing man hours, CPU time, testing effort testing coverage etc.

$$\tau \equiv s^\alpha u^{1-\alpha} \quad 0 \leq \alpha \leq 1$$ \hspace{1cm} (5.8)

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:

$$\Pr(N(s,u) = n) = \frac{(m(s,u))^n}{n!} \exp(-m(s,u)), \ n=0,1,2...$$  \hspace{1cm} (5.9)

and

$$m(s,u) = \int_{0}^{s} \int_{0}^{u} \lambda(\zeta, \xi) d\zeta d\xi$$  \hspace{1cm} (5.10)

### 5.2.4 S-Shaped Two-Dimensional Models: Unified Approach

In one dimensional analysis the object variable is dependent on one basic variable although the object takes on many different roles based upon its dependence on various other factors. Because a software failure is occurred when a input data hits to a software fault latent in a program, it is natural to say that important factors affecting the software reliability growth process are not only testing-time but also testing effort expenditure such as testing-coverage, the number of executed test-cases, and so on. Especially, the testing coverage is one of the important measures indicating the test adequacy and efficiency.

Here we develop two dimensional S-shaped model determining the combined effect of testing time and testing coverage.

Now we extend the testing time of one dimensional to a two dimensional problem considering testing resource as variable. Using the cobb-douglas production the corresponding mean value function is given as:

The mean value function with respect to testing resources is given as follows:

$$m(\tau) = a.F(\tau)$$  \hspace{1cm} (5.11)

Equation (5.11) is the generalized two-dimensional SRGM model (SRGMs). Substituting different types of distribution functions, we can obtain different mean value functions corresponding to them. The cumulative number of faults removed $m(\tau)$ is dependent on testing resources $\tau$. $\tau$ is a two-dimensional variable, with testing
time $s$ and testing coverage $u$ as its dimensions. The two-dimensional models are useful as they can show the effect of two aspects of a variable on which the result is dependent (Inoue and Yamada 2004, Wang et al 2011, Kapur et al 2010).

We define some additional notations as follows:

$m(s,u)$: mean number of faults removed corresponding to coverage $u$ and time $s$.

\[ m(s,u) = a.F(s,u) \]  
(5.12)

**Some Flexible SRGMs**

The Logistic distribution function can be used to model SRGM-1. The mean value function with respect to testing resources is given as follows:

\[ m(s,u) = a \cdot \frac{(1-\exp(-b_s^\alpha .u^{(1-\alpha)}))}{(1 + \beta .\exp(-b_s^\alpha .u^{(1-\alpha)}))} \]  
(5.13)

The Exponentiated Exponential distribution function can be used to model SRGM-2. The mean value function with respect to testing resources is given as follows (Ahuja and Nash 1967, Mucholkar and Srivastava 1993, Gupt and Kundu 2001):

\[ m(s,u) = a \cdot \left(1 - e^{-\phi.s^\alpha .u^{(1-\alpha)}}\right)^\omega \]  
(5.14)

The above mean value functions $m(\tau)$ represent the cumulative number of faults removed dependent on testing time $s$ and testing coverage $u$. Earlier many SRGMs have been discussed either it was time dependent, testing effort dependent or coverage dependent but here we develop a generalised framework for deriving two-dimensional SRGMs which show the joint effect of two metrics testing time and testing coverage to show the number of faults removed in the software.

**5.2.5 Data Set Description**

The data sets used for the two dimensional model are coverage data set. These data sets show the combined effect of testing time and testing coverage. We have used data set in
which cumulative numbers of faults found in data set 1 are 9 with execution of 796 test cases (Fujiwara and Yamada 2001) and 95.99% of test coverage.

5.2.6 Result of Parameter Estimation

The parameter estimation and comparison criteria results for data set of both models under consideration are given in Table 5.3 and 5.4 respectively. The fitting of the models to data set can be graphically illustrated. The testing resources are different for all the distributions as the value of $\alpha$, are estimated using the failure data. SRGM-2 has low Bias and MSE, depicting less fitting error. For SRGM-2 the real-time data set as adjusted $R^2$ is less than 0.992.

**TABLE-5.3**
Parameter Estimated values

<table>
<thead>
<tr>
<th>Parameters</th>
<th>a</th>
<th>b</th>
<th>$\beta$</th>
<th>$\phi$</th>
<th>$\varphi$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM-1</td>
<td>9.157</td>
<td>0.0214</td>
<td>0.092</td>
<td>-</td>
<td>-</td>
<td>0.532</td>
</tr>
<tr>
<td>SRGM-2</td>
<td>9.044</td>
<td>-</td>
<td>-</td>
<td>0.0189</td>
<td>0.588</td>
<td>0.79</td>
</tr>
</tbody>
</table>

**TABLE-5.4**
Comparisons Criteria

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Adj $R^2$</th>
<th>Bias</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM-1</td>
<td>0.991</td>
<td>0.009</td>
<td>0.0681</td>
</tr>
<tr>
<td>SRGM-2</td>
<td>0.992</td>
<td>-0.00771</td>
<td>0.0612</td>
</tr>
</tbody>
</table>
Figure 5.5: Goodness of fit for SRGM-1

Figure 5.6: Goodness of fit for SRGM-2