Chapter 3

Multi Release SRGMs for Fault Detection-Correction Processes and the effect of Reported Bugs

Software testing is the process of exercising a program with the specific intent of finding faults prior to delivery to the users. In some cases, fault correction is not performed immediately once a failure is detected. Every detected fault is reported, diagnosed, verified, and then corrected. The time between detection and correction is not negligible in real software testing process. The time to remove a detected fault depends on the complexity of the fault, the skill and experience of the debugging team, the available manpower, the software development environment and so on.

The idea of modeling fault correction processes as two-stage process was first proposed by Schneidewind (1975). Schneidewind described the testing process as a two-stage process in which all detected faults are corrected after a constant delay of time. Later on Xie and Zaho (1992) extended the Schneidewind model to a continuous version by using a time dependent delay function for the constant delay arguing that detected faults become harder to correct while testing is in progress.

Schneidewind (2001) extended his work by assuming that the time delay is a random variable following an exponential distribution. A general framework for modeling fault detection and correction processes in software reliability analysis was proposed

This Chapter is based on the following research papers entitled:


by Lo and Huang (2006) where some existing NHPP models were re-evaluated from the viewpoint of correction process. Further Xie, Hu, Wu and Ng (2007) discussed on fault correction process described by delayed detection process with a random or deterministic delay. As a result, combined fault detection and correction modeling presented more practical models for software testing process. All detection-correction modeling framework discussed in literature is based on single release of the software and none of them discuss about software with multiple releases. Here we proposed two stage detection and correction process for multi up-gradation of the software.

Later in the chapter, we discuss the effect of operational phase on FRP on successive release of the software. Apart from time, resource limitation, users need and some economic benefits from successive release (as discussed in chapter1 section 1.12), company plans up-gradation of software based on bugs reported from user side. By this policy, company can use bugs reported from the operational phase of previous releases to remove bugs from the old code of the current release of the software (Garmabaki et al., 2011).

This chapter is divided into two sections. In section 3.1 a unified framework for Fault Detection and Correction Processes for Successive release of software is proposed. In section 3.2 we incorporate the combined effect of bugs encountered during testing of present release and user reported bug from operational phase of the previous release in the model building. The model developed differentiate between the testing and the operational phase where fault removal phenomenon follows Kapur-Garg (1999) model and Weibull-model (Pham, 2006, Rausand and Hoyland, 2004) respectively. All developed models are validated on real dataset for software which has been released in the market four times.

3.1 Unification Scheme in Multi Up- Gradation Software Reliability Incorporating Detection and Correction Process

In this section a multi up-gradation SRGM incorporating delayed detection-correction process has been developed. This model uniquely identifies the faults left undetected in the software release presently in operation and which are removed subsequently during testing of the next release. We propose a generalized framework for fault
detection and correction in multi up-gradation software system. The models developed have been validated and verified using real data sets.

3.1.1 Assumptions:

The basic assumptions of the model are as follows:

1. The fault detection/correction process are modeled by NHPP.
2. The number of faults detected at any time is proportional to the remaining number of faults in the software.
3. The number of faults in the beginning of the testing phase is finite.
4. All faults are mutually independent from failure detection point of view.
5. Fault removal process is perfect.

3.1.2 Notations

\[ m(t) \] : Expected number of faults removed by time \( t \).
\[ f(t), g(t) \] : Probability density function.
\[ F(t), G(t) \] : Distribution functions for fault detection and fault correction process.
\[ \Gamma(t, \alpha, \theta) \] : Gamma distribution function.
\[ t_i \] : Time for \( i^{th} \) release \( i=1..4 \).
\[ a_i \] : Fault content of \( i^{th} \) release \( i=1..4 \).
\[ a \] : Total Initial fault content.
\[ \otimes \] : Steiltjes convolution operator.
\[ \ast \] : Convolution operator.
\[ \beta_i \] : Logistic learning factor for \( i^{th} \) release, \( i=1..4 \).
\[ b_i \] : Fault detection rate for Kapur/logistic model for \( i^{th} \) release \( i=1..4 \).
\[ \beta, b \] : Constant parameter.
\[ \alpha_i \] : scale parameter and \( \alpha_i > 0 \).
\[ \theta_i \] : shape parameter and \( \theta_i > 0 \).
3.1.3. Generalized multi release SRGM with Detection and Correction as Two Stage Process

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

Based on the NHPP assumptions, it may be shown that \( \{N(t), t \geq 0\} \) has Poisson distribution with mean \( m(t) \), i.e.,

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

(3.1)

By definition, the mean value function of cumulative number of fault, \( m(t) \), can be expressed in terms of the fault intensity function of the software, i.e.,

\[
m(t) = \int_0^t \lambda(s) ds
\]  

(3.2)

Let ‘\( a \)’ denote the expected number of faults that would be detected given infinite testing time in case of finite failure NHPP models. Then, the mean value function of the finite failure NHPP models can also be written as:

\[
m(t) = aF(t)
\]  

(3.3)

Where \( F(t) \) is a distribution function.

Now, in order to incorporate the two stage testing processes i.e. failure detection process followed by fault correction in a unified or generalized modeling framework, Equation (3.3) may be modified as (which may be proved on the lines of (Musa et al., 1987)):

\[
m(t) = a(F \otimes G)(t)
\]  

(3.4)

Hence,
\[
\frac{dm(t)}{dt} = a(f^*g)(t) \quad (3.5)
\]

The above Equation (3.5) can be rewritten as follows:

\[
\frac{dm(t)}{dt} = \frac{(f^*g)(t)}{[1-(F \otimes G)(t)]}[a-m(t)] \quad (3.6)
\]

\[
\frac{dm(t)}{dt} = s(t)[a-m(t)]
\]

Where \( s(t) = \frac{(f^*g)(t)}{[1-(F \otimes G)(t)]} \) is the failure observation/detection-fault removal/correction rate.

### 3.1.4. Modeling Multi Up-Gradation Framework

#### 3.1.4.1. Modeling for Release 1

Let the First Release of software be done at \( t = t_1 \). Note that this model classifying fault removal process into two processes as fault detection and correction processes. Also we can’t remove all faults in each release and some of fault remained in the code when we release software. The mathematical equation of these finite numbers of faults removed is given as:

\[
m_1(t) = a(F_i \otimes G_i)(t) \quad \quad 0 \leq t \leq t_1 \quad (3.7)
\]

Where \( F_i(t) \) and \( G_i(t) \) are distribution functions for detection and correction process, respectively.

#### 3.1.4.2. Modeling for Release 2

The scenario for the modeling of second release is not same as first releases. The modeling framework discuss the removal of the fault arising due to new functionality added to the software as well as by those faults which remained undetected during the testing of previous release. The leftover undetected fault content of the first release
may calculated by \(a_1[1-(F_1 \otimes G_1)(t_1)]\). It may be noted that \((F_2 \otimes G_2)(t-t_1)\)
represents fraction of fault detected/corrected during release 2.

\[
m_2(t) = a_2(F_2 \otimes G_2)(t-t_1) + a_1[1-(F_1 \otimes G_1)(t_1)](F_2 \otimes G_2)(t-t_1) \quad t_1 < t \leq t_2
\]
\[
= [a_2 + a_1[1-(F_1 \otimes G_1)(t_1)]](F_2 \otimes G_2)(t-t_1) \quad (3.8)
\]

Where \(F_2(t)\) and \(G_2(t)\) are distribution functions as detection and correction process of release 2.

3.1.4.3. Modeling for Release 3 and 4

We may describe the FRP for release 3 and 4 as the previous subsection. Mathematical equations may be represented as follows:

\[
m_3(t) = a_3(F_3 \otimes G_3)(t-t_2) + a_2[1-(F_2 \otimes G_2)(t_2-t_1)](F_3 \otimes G_3)(t-t_2) \quad t_2 < t \leq t_3
\]
\[
= [a_3 + a_2[1-(F_2 \otimes G_2)(t_2-t_1)]](F_3 \otimes G_3)(t) \quad (3.9)
\]

\[
m_4(t) = a_4(F_4 \otimes G_4)(t-t_3) + a_3[1-(F_3 \otimes G_3)(t_3-t_2)](F_4 \otimes G_4)(t-t_3) \quad t_3 < t \leq t_4
\]
\[
= [a_4 + a_3[1-(F_3 \otimes G_3)(t_3-t_2)]](F_4 \otimes G_4)(t-t_3) \quad (3.10)
\]

Where \(F_3(t-t_2)\), \(G_3(t-t_2)\), \(F_4(t-t_3)\) and \(G_4(t-t_3)\) are defined as in previous subsections.

3.1.5. Derivation of New and Existing Models

In this section, we will derive two MUSRGMs by using proposed model framework for release 1 to 4.

3.1.5.1. Multi Up-Gradation Model, MUSRGM 1

In the first SRGM, we use logistic distribution for representing detection process and identify function as correction process. Kapur and Garg model (1999) is one of most
popular model in software reliability area and can fit on many types of failure data because of flexible nature which related to learning parameter $\beta$. For deriving model for release 1, let us consider that:

\[
\begin{align*}
F_1 &: T \sim \text{Log.dis.}(b_1, \beta_1) \\
G_1 &: T \sim 1(t)
\end{align*}
\]

Then

\[
m_1(t) = (F_1 \otimes G_1)(t) = \left[ 1 - \left( \frac{(1 + \beta_1)e^{-b_1t}}{1 + \beta_1 e^{-b_1t}} \right) \right] ; \quad 0 \leq t \leq t_1
\]  

(3.11)

The corresponding mathematical expression for release 2 to 4 may be given by:

\[
\begin{align*}
F_i &: T \sim \text{Log.dis.}(b_i, \beta_i) \\
G_i &: T \sim 1(t) ; \quad i = 2, 3, 4
\end{align*}
\]

\[
m_i(t) = [a_2 + a_i(1 - F_1 \otimes G_1)(t_1)].(F_i \otimes G_i)(t - t_1)
\]

\[
= [a_2 + a_1 \cdot \left( \frac{(1 + \beta_1)e^{-b_1(t_1)}}{1 + \beta_1 e^{-b_1(t_1)}} \right)] \left[ 1 - \left( \frac{(1 + \beta_2)e^{-b_2(t_2)}}{1 + \beta_2 e^{-b_2(t_2)}} \right) \right]
\]

\[
t_1 < t \leq t_2 \quad (3.12)
\]

\[
m_i(t) = [a_3 + a_2(1 - F_2 \otimes G_2)(t_2 - t_1)].(F_i \otimes G_i)(t - t_2)
\]

\[
= [a_3 + a_2 \cdot \left( \frac{(1 + \beta_2)e^{-b_2(t_2 - t_1)}}{1 + \beta_2 e^{-b_2(t_2 - t_1)}} \right)] \left[ 1 - \left( \frac{(1 + \beta_3)e^{-b_3(t_3 - t_2)}}{1 + \beta_3 e^{-b_3(t_3 - t_2)}} \right) \right]
\]

\[
t_2 < t \leq t_3 \quad (3.13)
\]

\[
m_i(t) = [a_4 + a_3(1 - F_3 \otimes G_3)(t_3 - t_2)].(F_i \otimes G_i)(t - t_3)
\]

\[
= [a_4 + a_3 \cdot \left( \frac{(1 + \beta_3)e^{-b_3(t_3 - t_2)}}{1 + \beta_3 e^{-b_3(t_3 - t_2)}} \right)] \left[ 1 - \left( \frac{(1 + \beta_4)e^{-b_4(t_4 - t_3)}}{1 + \beta_4 e^{-b_4(t_4 - t_3)}} \right) \right]
\]

\[
t_3 < t \leq t_4 \quad (3.14)
\]
Note that, we avoid substituting all recursive function into the above relation and used general form for representation.

3.1.5.2. Multi Up-Gradation Model, MUSRGM 2

In the second SRGM, we used exponential distribution as detection process and Gamma distribution as correction process. In probability theory and statistics, the Gamma distribution is a two-parameter family of continuous probability distributions (Pham, 2006, Rausand and Hoyland, 2004). It has a scale parameter \( \alpha \) and a shape parameter \( \theta \). The Gamma distribution function is flexible model that can empirically fit many types of failure data. Let consider that

\[
\begin{align*}
  F_i &= \exp(b t) \\
  G_i &= \text{Gamma}(\alpha, \theta)
\end{align*}
\]

The cumulative number of fault removed for release 1 is given by:

\[
m_1(t) = a_1 (F_1 \otimes G_1)(t) \\
= a_1 \left( \Gamma(t, \alpha_1, \theta_1) - \frac{\exp(-b t)}{(1-b \theta_1)^{\alpha_1}} \times \Gamma(t, \alpha_1, \frac{\theta_1}{1-b \theta_1}) \right); \quad 0 \leq t \leq t_1
\]

In the same way we may drive mathematical expression for each release by using relation (3.7) to (3.10).

3.1.6. Data Set and Model Validation

Validity of the proposed models have been tested on Tandem computer (Wood, 1996, Pham, 2006) four release data set as pointed in chapter 2 section 2.1.5.

The Performance analysis of proposed model is measured by the five common criteria as MSE, BIAS, MRSPE, \( R^2 \) and Variation(Pham, 2006, Kapur et al., 2011e).

3.1.7 Parameter Estimation and Goodness of Fit and Data Analysis
Estimated value of parameters of each releases are given in Table 3.1(a) and Table 3.1(b) for MUSRGM-1 and MUSRGM-2.

In addition, the five goodness of fit measure such as MSE, $R^2$, Bias, RMSPE and Variation are summarized in Table 3.2(a) and Table 3.2(b) for four successive releases of software system related to MUSRGM-1 and MUSRGM-2.

The value of $R^2$ (MSE) for MUSRGM-1 in all releases is higher (lower) than that MUSRGM-2 and provides better goodness of fit for all releases (For more details see Table 3.2(a) and Table 3.2(b)).

**Table 3.1(a):** Parameter estimates for MUSRGM-1

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Releases</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$a_i$</td>
<td>111</td>
<td>121</td>
<td>62</td>
<td>43</td>
</tr>
<tr>
<td>$b_i$</td>
<td>0.172</td>
<td>0.254</td>
<td>0.483</td>
<td>0.262</td>
</tr>
<tr>
<td>$\beta_i$</td>
<td>1.204</td>
<td>3.788</td>
<td>11.16</td>
<td>5.739</td>
</tr>
</tbody>
</table>

**Table 3.1(b):** Parameter estimates for MUSRGM-2

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Releases</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$a_i$</td>
<td>125</td>
<td>125</td>
<td>69</td>
<td>43</td>
</tr>
<tr>
<td>$b_i$</td>
<td>0.09803</td>
<td>0.16295</td>
<td>0.29459</td>
<td>0.16525</td>
</tr>
<tr>
<td>$\alpha_i$</td>
<td>0.08427</td>
<td>0.551141</td>
<td>1.37018</td>
<td>1.067561</td>
</tr>
<tr>
<td>$\theta_i$</td>
<td>0.04523</td>
<td>0.07226</td>
<td>0.27362</td>
<td>0.15324</td>
</tr>
</tbody>
</table>
Table 3.2(a): Goodness of fit criteria for MUSRGM-1

<table>
<thead>
<tr>
<th>Criteria</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R^2$</td>
<td>0.989</td>
<td>0.995</td>
<td>0.993</td>
<td>0.995</td>
</tr>
<tr>
<td>BIAS</td>
<td>0.435</td>
<td>0.34</td>
<td>-0.1011</td>
<td>-0.1185</td>
</tr>
<tr>
<td>MSE</td>
<td>8.979</td>
<td>6.001</td>
<td>3.1547</td>
<td>0.9774</td>
</tr>
<tr>
<td>RMSPE</td>
<td>3.0727</td>
<td>2.5156</td>
<td>1.8556</td>
<td>1.0146</td>
</tr>
<tr>
<td>Variation</td>
<td>3.0417</td>
<td>2.4925</td>
<td>1.8529</td>
<td>1.0077</td>
</tr>
</tbody>
</table>

Figure 3.1 and Figure 3.2 show the estimated and the actual values of the number of faults removed for four releases of MUSRGM1 and MUSRGM2.
Figure 3.1: Goodness of fit of releases 1 to 4 due to (MUSRG M2)

Table 3.2(b): Goodness of fit criteria for MUSRM-2

<table>
<thead>
<tr>
<th>Criteria</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R^2$</td>
<td>0.986</td>
<td>0.991</td>
<td>0.984</td>
<td>0.995</td>
</tr>
<tr>
<td>BIAS</td>
<td>0.22604</td>
<td>0.30391</td>
<td>0.44468</td>
<td>0.05298</td>
</tr>
<tr>
<td>MSE</td>
<td>11.3414</td>
<td>12.2676</td>
<td>7.3639</td>
<td>1.00377</td>
</tr>
<tr>
<td>RMSPE</td>
<td>3.4580</td>
<td>3.5959</td>
<td>2.8311</td>
<td>1.02789</td>
</tr>
<tr>
<td>Variation</td>
<td>3.4474</td>
<td>3.6087</td>
<td>2.7960</td>
<td>1.02926</td>
</tr>
</tbody>
</table>
3.2. Development of a Multi-Release SRGM Incorporating the Effect of Bugs Reported from Operational Phase

In this section, we discuss about testing and operational phases of the software. In successive release policy, scenario for SDLC is not the same as the single release of the software.

In previous models, the effect of leftover fault of previous release are discuss on the FRP by incorporating imperfect debugging and fault of different severity. Also we assumed that all leftover fault of previous release will remove by testing team without enhancing of reported bugs from the operational phase. In this section, the effect of reported bugs is considered on the model building.

During testing of multi release software, it is quite possible that some faults of old code be removed by testing team of new release (without using any bug reports of operational phase of the previous release of the software) and some others be removed on the basis of bug reported during operational phase.
The relation between testing phase and operational phase of each release has been depicted in Figure 3.3. In this section, we developed modeling framework to represent the combined effect of bugs encountered during testing of present release and user reported bug from operational phase. The developed model takes into consideration the testing and the operational phase where fault removal phenomenon follows Kapur-Garg model and Weibull-model, respectively (Kapur et al., 1999, Rausand and Hoyland, 2004). Later in the chapter, the model is validated on real software dataset.

3.2.1. Assumptions:

The basic assumptions of the model are as follows:

1) The fault removal process for each release is modeled by NHPP.

2) The number of faults detected at any time is proportional to the remaining number of faults in the software.

3) The undetected faults of previous release are removed during testing of new release either on the basis of failure observed during testing or due to bugs reported by the user from operational phase.
4) The number of faults in the beginning of the testing phase is finite.
5) All faults are mutually independent from failure detection point of view.
6) Each time a failure occurs, the fault that caused it is immediately fixed, and no new faults are introduced.

### 3.2.2. Notations

The following notation used in the paper.

- \( m(t) \) Expected number of faults removed by time \( t \).
- \( \lambda(t) \) Failure intensity.
- \( F(t) \) Probability distribution functions for FRP
- \( F_i^{test}(t), F_i^{op}(t) \) Probability distribution functions for FRP during testing phase and operational phase, respectively.
- \( \tau_i \) Time for \( i^{th} \) release, \( i=1..n \).
- \( a_i \) Initial fault content for \( i^{th} \) release, \( i=1..n \)
- \( a \) Total fault content in the software.
- \( b(t) \) Time dependent fault detection rate function.
- \( \beta_i \) Logistic learning factor in Kapur-Garg model for \( i^{th} \) release, \( i=1..n \).
- \( b_{test} \) Fault detection rate during testing phase.
- \( b_{op} \) Fault detection rate during operational phase.
- \( \beta,b \) Constant.
- \( \lambda \) Proportion of old faults\(^*\) removed by testing team.
- \( 1-\lambda \) Proportion of old faults\(^*\) removed based on reported bugs.

\(^*\) old faults are undetected faults from previous release.

### 3.2.3. Testing Phase in Software Development Life Cycle
Software testing is very important stage of SDLC and has direct influence on operational phase and reliability of software system. Software testing has three main purposes: verification, validation, and failure finding which was described in chapter 1 sections 1.4 and 1.6.

**Kapur-Garg model For Testing Phase**

Logistic model is one of the most popular models in software reliability area and can fit on many types of failure data in testing phase because of its flexible nature. This model has been discussed in the chapter 1, section 1.9.1.4.

Here, we use following model to represent during testing of successive release of software.

\[
m(t) = a \cdot \frac{1 - e^{-\beta t}}{1 + \beta e^{-\alpha t}} = a F^{op}(t)
\]  

(3.16)

### 3.2.4. Operational Phase in Software Development Life Cycle

The final phase in software lifecycle is operational phase and one of the important activities in this phase is maintenance. Maintenance is defined as any change made to the software, either to correct a deficiency in its performance to compensate for environmental change, or to enhance its operation (Pham, 2006).

In this model, we try to capture the effect of reported bugs from operational phase related to previous release of the software. We have used Weibull distribution function for fault removal process in operational phase as explained in the next subsection.

**The Weibull Model for Operational Phase**

In probability theory and statistics, the Weibull distribution belongs to two-parameter family of continuous probability distributions. It has a scale parameter \( \beta \) and a shape parameter \( \alpha \). The probability distribution and density function are given as:
Chapter 3  
Multi Release SRGMs for Fault Detection-Correction Processes and the effect of Reported Bugs

\[ F^{op}(t) = (1 - e^{-b_{op} \cdot t}) \]  

(3.17) \[ f^{op}(t) = k \cdot b_{op} \cdot t^{k-1} \cdot e^{-b_{op} \cdot t} ; k > 0, b_{op} > 0 \]

The Weibull distribution function is flexible that can empirically fit many types of failure data and has parameters that can account for different increasing and decreasing trends in the failure rate. It reflects initial increase and eventual decrease in fault/failure occurrences, which can vary across multiple releases. We would generally expect early failure occurrences to reflect an increasing trend as users switch to the present release, executed the software, and report bugs. Later failure occurrences reflect a decreasing trend.

Intuitively, the Weibull density function can be broken down into two parts:

- an increasing component \( k \cdot b_{op} \cdot t^{k-1} \) which has more effect in the early stage of process,
- a decreasing component \( e^{-b_{op} \cdot t} \), which dominates as time increases.

In general, \( k \cdot b_{op} \cdot t^{k-1} \) increases as a function of time and can account for increases in fault/failure occurrences. The rate of growth is controlled by a combination of the \( k \) and \( b_{op} \) parameters. The increasing component is flexible enough to describe both concave and convex increasing patterns. Concave increasing patterns can occur when the growth in failure occurrences is faster at the beginning of a release, which may occur if many users quickly adopt and use a release. Convex increasing patterns can occur when failure occurrences increase slowly. This may occur if users slowly migrate to the current release or if constraints on development and problematic content cause blocking which prevent faults from being discovered.

The term \( e^{-b_{op} \cdot t} \) decreases as a function of time and can account for decreases in fault/failure occurrences. Again, the rate of decrease is controlled by a combination of the \( k \) and \( b_{op} \) parameters. The decreasing component can describe concave or convex decreasing patterns. Convex decreasing patterns can occur when failure occurrences decrease rapidly, which may occur if there is fast migration to a new release. Concave
patterns can occur when the failure occurrences decrease slowly, which may occur if failure occurrences remain high over a longer time period due to constraints on development or problematic content. It is reasonable to develop a flexible model such as Weibull model which can fit failure data well. The Exponential, the Power, and the Logarithmic models do not have both decreasing and increasing components, thus cannot describe the interplay of increasing and decreasing trends (Li et al., 2004).

Here, we use Weibull distribution to model fault removal corresponding to bug reported by user. In subsequent section we propose modeling framework for multiple release software on the basis of discussion done in this section.

### 3.2.5. Multiple Release Model Development

The proposed model is based on the assumption that the overall fault removal of the new release depends on the reported faults from the just previous release of the software and on the faults generated due to adding some new functionalities (addons/up-gradations) to the existing software system.

In practice, it is important to know that how many faults exist in the software at any time, so that different testing strategy and testing effort can be applied to remove those faults. We can define different scenario for testing and operational phases for successive releases of the software and use a variety of mathematical expressions for modeling fault removal process.

#### 3.2.5.1. Modeling for Release 1

Let consider that testing begin at time $t = 0$ and the first release of software be done at $t = \tau_1$. Note that we can’t remove all faults during testing phase and some of fault remain in the code even after its release. The mathematical equation of these finite fault count model is given as:

$$ m_i(t) = a_i F_{\tau_1}^{test}(t), \quad 0 \leq t \leq \tau_1 $$

Note that during the testing phase of first release no bug report is available. So the faults are removed on the basis of testing only.
Chapter 3

Multi Release SRGMs for Fault Detection-Correction Processes and the effect of Reported Bugs

3.2.5.2. Modeling for Release 2

Due to fierce competition and technological changes, the software developer is forced to add new features to the software. New features added to the software leads to complexity and increase in the fault content of the software. While testing the newly formed code, there is always a possibility that the testing team may find some faults which were present in previously developed code. Testing the newly developed code helps the developer to actually improve the software overall as it also removes some faults of previously developed code. During testing, it is quite possible that some of faults of old code is removed directly by testing team of new release (without using any bug reports of operational phase of the previous release of software) and some others are removed on the basis of bug reported during operational phase. In addition due to parallel testing, several people may report the same fault/failure which we call duplicates. Duplicates are not included in count of unique fault. Based on above framework, we may write following mathematical equation for second release.

\[
m_2(t) = a_2 . F^{\text{est}}_2(t - \tau_1) + \lambda_i . (a_i - m_i(\tau_1)) . F^{\text{est}}_2(t - \tau_1) \\
+ (1 - \lambda_i) . (a_i - m_i(\tau_1)) . F^{\text{op}}_1(t - \tau_1) \\
= \left[ a_2 + \lambda_i . (a_i - m_i(\tau_1)) \right] . F^{\text{est}}_2(t - \tau_1) \\
\left[ (1 - \lambda_i) . (a_i - m_i(\tau_1)) \right] . F^{\text{op}}_1(t - \tau_1)
\]

(3.19)

Where \( \tau_1 < t \leq \tau_2 \).

At this step, faults generated due to enhancement of the features are removed with distribution function \( F^{\text{est}}_2(t) \) and \( \lambda_i . (a_i - m_i(\tau_1)) \) represent some of undetected faults of the previous release which interacts with new portion of code and are removed/detected by testing team of second release (without using any bug reported during operational phase of the previous release) i.e. \( F^{\text{est}}_2(t - \tau_1) \).

\( 1 - \lambda_i ) . (a_i - m_i(\tau_1)) . F^{\text{op}}_1(t) \) are the remaining undetected fault related to previous release which are removed on the basis of bug reported throughout operational phase of release 1.
3.2.5.3. Modeling for Release $n$

It may be noted that because of limited time and resources, some of faults due to new feature remain in the code when we release next version of the software at time $t = \tau_{n-1}$ . Same situation will happen on the $n^{th}$ releases of the software.

$$m_n(t) = [a_n + \lambda_{n-1}](a_{n-1} - m_{n-1}(\tau_{n-1} - \tau_{n-2}))].F_{n}^{\text{test}}(t - \tau_{n-1})$$

$$+[(1-\lambda_{n-1})].(a_{n-1} - m_{n-1}(\tau_{n-1} - \tau_{n-2}))].F_{n-1}^{\text{op}}(t - \tau_{n-1})$$

(3.20)

Where $\tau_{n-1} < t \leq \tau_n$ and $F_{n}^{\text{test}}(t - \tau_{n-1})$ , $F_{n-1}^{\text{op}}(t - \tau_{n-1})$ are Kapur-Garg model and Weibull model respectively. For simplifying the model we introduce new formulation as:

$$m_n(t) = m_n^{\text{test}}(t) + m_n^{\text{op}}(t)$$

(3.21)

It may be noted that some of the faults remain in the code when we release software at time $t = \tau_n$ . It is reasonable that in initial releases of the software, testing team receives less number of reports in comparison to the later releases because the less number of users adopt the software in initial releases and as the number of users increases, the number of reports may also increase in subsequent releases. This increase does not have a monotonic behavior i.e. every future releases may not prove successful in the market. For example a lot of users bought Windows 98 but didn’t adopt next release of Windows (namely Windows Millennium).Same situation happened for Windows XP and Vista also(wikipedia.org, 2012).

3.2.6. Data Set and Model Validation

To verify the proposed model that incorporates both effect of testing and operational phase on the removal process, we use Tandem computers data set (Wood, 1996) which describe in chapter 2 section 2.1.5.

The performance analysis of proposed model is done by the five common criteria that we define as $R^2$, Bias, Variation, RMSPE, MSE (Pham, 2006, Kapur et al., 2011). The proposed model give very good fit as exhibited by the values of various goodness of fit criteria.
3.2.7. Parameter Estimation and Goodness of Fit

The parameters of all releases are estimated and the related mean value functions are obtained. Here we estimate the parameters $a$, $b_{\text{test}}$ and $\beta$ for testing phase and $b_{\text{op}}, k$ for operational phase. Estimated value of parameters for model in each release are given in Table 3.3(a). In addition, Table 3.3(b) shows the values of goodness of fit related to proposed model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>111</td>
<td>123</td>
<td>69</td>
<td>44</td>
</tr>
<tr>
<td>$b_{\text{test}}$</td>
<td>0.172</td>
<td>0.245319</td>
<td>0.506165</td>
<td>0.245238</td>
</tr>
<tr>
<td>$b_{\text{op}}$</td>
<td>-----</td>
<td>0.162383</td>
<td>0.125603</td>
<td>0.292849</td>
</tr>
<tr>
<td>$\beta$</td>
<td>1.204</td>
<td>3.631556</td>
<td>13.61768</td>
<td>4.305296</td>
</tr>
<tr>
<td>$k$</td>
<td>-----</td>
<td>0.838084</td>
<td>1.148504</td>
<td>0.267752</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>-----</td>
<td>0.911529</td>
<td>0.656427</td>
<td>0.54252</td>
</tr>
</tbody>
</table>

Table 3.3(a): Parameter estimates
Table 3.3(b): Goodness of fit criteria

<table>
<thead>
<tr>
<th>Criteria</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R^2$</td>
<td>0.989</td>
<td>0.995</td>
<td>0.996</td>
<td>0.995</td>
</tr>
<tr>
<td>Bias</td>
<td>0.43528</td>
<td>0.308622</td>
<td>0.26824</td>
<td>0.00392</td>
</tr>
<tr>
<td>MSE</td>
<td>8.979214</td>
<td>5.970414</td>
<td>1.89693</td>
<td>0.88140</td>
</tr>
<tr>
<td>Variation</td>
<td>3.04177</td>
<td>2.490294</td>
<td>1.38366</td>
<td>0.96455</td>
</tr>
<tr>
<td>RMSPE</td>
<td>3.07275</td>
<td>2.509345</td>
<td>1.409424</td>
<td>0.96455</td>
</tr>
</tbody>
</table>

Also Figure 3.4 to Figure 3.7 shows the estimated and the actual values of the number of faults removed for each releases, separately.

Figure 3.4: Goodness of fit of release 1
Figure 3.5: Goodness of fit of release 2

Figure 3.6: Goodness of fit of release 3
3.2.8. Data Analysis

Based on the estimation result shown in Table 3.3(a), we may conclude that during initial releases there are less number of reported bugs. It may be attributed to less number of users initially, and the reported bug from operational phase increase when time increase but with different rate for all four releases. Figure 3.8 presents the rate of growth of reported bugs.

Figure 3.7: Goodness of fit of release 4

Figure 3.8: The rate of growth of reported bug from operational field
It may be noted that in the actual failure data, the value of MSE without consideration of operational phase for release 1 to 4 are compared to the MSE of proposed model ,[refer to Table 3.4].

**Table 3.4**: Comparison of MSE criteria .

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Releases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>MSE with Operational phase</td>
<td>8.979</td>
</tr>
<tr>
<td>MSE without Operational phase</td>
<td>8.979</td>
</tr>
</tbody>
</table>