CHAPTER 2

RELIABILITY FACTOR APPROACH FOR UNCERTAINTY ANALYSIS IN SOFTWARE DEVELOPMENT

2.1 INTRODUCTION

The goal of an industry is to develop error and fault free software. That software is delivered to the market with good quality. Software reliability is the important factors of software quality. Reliability engineering is an important aspect of many system development efforts and consequently there has been a great deal of research in the software based systems Huang and Lin (2006). One important activity included in reliability engineering is reliability prediction and improvement. This chapter deals with reliability improvement using growth modeling techniques.

Software reliability plays a vital role in the improvement of quality of a software. This chapter presents software growth models. Section 2.2 presents improved Bayesian Approach for uncertainty analysis. Section 2.3 presents Software Reliability models based on Logistic Effort Test Function.
2.2 IMPROVED BAYESIAN APPROACH FOR UNCERTAINTY ANALYSIS

In other software reliability models, the parameters are typically estimated from the test data of the corresponding component. However, the widely used point estimators are subject to random variations in the data, resulting in uncertainties in these estimated parameters. Ignoring the parameter uncertainty can result in grossly underestimating the uncertainty in the total system reliability to apply the models for predicting the reliability of the component; the parameters of the models need to be known or estimated. Field data or data from components with similar functionalities are usually available to help estimate these parameters, but the estimators are subject to random variation because they are functions of random phenomena. Parameter uncertainty arises when the input parameters are unknown. Moreover, the reliability computed from the models, which are functions of these parameters, is not sufficiently precise when the parameters are uncertain.

Reliability is the probability of a device performing its purpose adequately for the period intended under the given operating conditions. Challenges in software reliability are the lack of available failure data from a single test, which often makes modeling difficult. Further, the data poses a bigger challenge in the uncertainty analysis of the software reliability modeling. In this chapter, the improved Bayesian statistical method and Logistic test effort model are proposed to quantify the uncertainties in the software reliability model of system with correlated parameters.
2.2.1 System Architecture

The architecture of the proposed system is depicted in the Figure 2.1. The description of uncertainty analysis using Bayesian approach. Bayesian modeling is a consistent formal modeling framework based on Bayesian statistics. In Bayesian statistics all unknown parameters of the system are considered as random parameters. The uncertainties of the unknown parameters are expressed with prior probability distributions. To make inference on the parameters, the links between the parameters and data should be described. The Principle of Maximum Entropy is used to discover the probability distribution which leads to the highest value for this uncertainty, thereby assuring that no information is inadvertently assumed. The description of uncertainty analysis is done using the Bayesian approach. The failure data is collected according to the failure analysis by expert’s knowledge. That dearth data is converted to a process data using maximum entropy principle. Quantified parameter is used for measuring uncertainty analysis using Bayesian approach. Co-efficient rank is fixed for validating those parameters. The parameter is adjusted according to the rank to improve the performance.

Figure 2.1 Uncertainty Analysis using Bayesian Approach
Figure 2.2 (Continued)
2.2.2 Methodology

2.2.2.1 Reliability factor approach

Reliability is the probability of a device performing its purpose adequately for the period intended under the given operating conditions. The definition brings into focus four important factors namely,

i. The reliability of a system is expressed as a probability.

ii. The system is required to give adequate performance.

iii. The duration of adequate performance is specified.

iv. The environmental or operating conditions are prescribed.
Reliability is usually measured in terms of probabilities. The theory of Bayesian statistics is a well established and the method has been applied in various areas including software, automation systems, medical diagnosis, geological explorations etc. In the software based system, the uncertain variables are associated to each component, where the uncertainty is expressed by probability density. The probability density expresses belief or confidence in the various possible outcomes of variable. This probability depends conditionally on the status of other components based on different input.

The probabilities are estimated by means of statistical methods. Various statistical methods are available for estimating the reliability of the system, but these methods are not suitable for estimating the reliability of software based system, because, these methods have not considered the uncertainties of unknown parameters. The Bayesian statistical method considers the uncertainties of unknown parameters. So, Bayesian model is mainly used for estimating the reliability of software-based systems. This method is also used to predict the future of the system by the observed information. Reliability is the probability that a system will operate without failure for a given time in a given environment.

Note that reliability is defined for a given environment. Since the test and operations environments are generally different, model results from test data may not apply to an operations environment. The "given time" in this definition may represent any number of actual data items, such as number of executions; number of lines of code traversed, or wall clock time. The use of a model also requires careful definition of what a failure is. Reliability models can be run separately on each failure type and severity level. Models have
been developed to measure, estimate and predict the reliability of computer software.

Software reliability has received much attention because reliability has always had obvious effects on highly visible aspects of software development: testing prior to delivery, and maintenance Wooff et al (2002). Early efforts focused on testing primarily because that is when the problems appeared. As technology has matured, the root causes of incorrect and unreliable software have been identified earlier in the life cycle. This has been due in part to the availability of results from measurement research and/or application of reliability models.

2.2.2.2 Entropy

If any of the probabilities is equal to 1 then all the other probabilities are 0 and then know exactly which state the system is in. Since probabilities are used to cope with lack of knowledge, and since one person may have more knowledge than the other, it follows that two observers may, because of their different knowledge, use different probability distributions Eyink and Kim (2005). In this sense probability, and all quantities that are based on probabilities, are subjective. Uncertainty is expressed quantitatively the information which does about the state of the information. This information is

\[ K_{ij} = \sum p(T_i + T_j) \log_2 \left( \frac{1}{p(T_i) + j(T_i)} \right) \]  

\[ L_{ij} = \sum (p(T_i + T_j) \ast g(T_i + T_j)) \]
Information is measured in bits because it uses logarithms with base 2. A person may have different information about the system from another person, and therefore would calculate a different numerical value for entropy. The Principle of Maximum Entropy is used to discover the probability distribution which leads to the highest value for this uncertainty, thereby assuring that no information is inadvertently assumed. The system reliability can be evaluated using the architecture and relationship of the components.

2.2.2.3 Assumptions

Complicated software contains multiple modules. Many tools can be implemented to evaluate the system reliability, given the parameters of modules. Regardless of the tools used, the system reliability is a function combining the parameters of its components. As a result, the uncertainties in the parameters affect the whole system reliability. The purpose of this section is to study and quantify the uncertainty in the reliability of the complex system, due to the uncertainty of the parameters in the numerous components of the system. Some general assumptions of the system-level analysis are listed as follows:

1. Consider system contains different modules.

2. Each module has its own reliability model and let \( j \) denote the set of parameters for the \( i \)th module’s model, where \( j = 1; 2 \ldots K \).

3. For each module, the probability distribution of the model’s parameters is known, which can be derived using formula 2 and 4.
4. The failures of different components are ‘d’ independent, the system reliability can be calculated by the function of modules Parameters as \( d = f (v_1, v_2, \ldots, v_j) \).

Based on the above assumptions, an MC simulation is presented for generally analyzing the uncertain system reliability. It is difficult to use analytic methods for combining the distributions of numerous parameters to derive the probability density function of the system reliability, especially for complicated systems with complex architecture and many components. Hence, the MC simulation becomes a practical way to make the uncertainty analysis of the complicated system tractable.

2.2.2.4 Extracting Data after implementing MEP

To combine MEP with BA for the uncertainty analysis, it is important to first extract data from the experts, and then the history by using the MEP and then input them into the prior distributions of BA. The goal of MEP is to incorporate all available information, including the unknown information which are available in both discrete and continuous distribution.

a) Discrete Distribution

Consider the case when an expert gives some information about a simple constraint, for example, \( p_1 + p_2 = 0.3 \), for a discrete distribution. Then, the distribution (probability mass function (pmf)) with ME is \( p_1 = p_2 = 0.15 \) and the rest \( p_i = 1 - 0.3 / n - 2 \) (\( i = 3, 4, \ldots, n \)). Alternatively, if it is provided with information on the mean values \( F_k \) of a certain function \( f_k(x) \) of data, then this information can be expressed as \( m \) constraints

\[
f_{k(x)} = \sum_{i = 1}^{n} Pr(x_i \mid I) f_k(x_i) = F_k \quad k = 1, \ldots, m \quad (2.3)
\]
where \( \Pr(x_i | I) \) denotes the probability for each possible state \( i \), given the information.

### 2.2.3 System Modules

#### 2.2.3.1 Measurement of uncertainty analysis

After deriving the priori distribution from MEP and observing failure times ‘s’, the posterior distribution can be obtained. Then, the marginal density function with respect to each parameter can be obtained by the integration formula. In addition, the mean value of the corresponding parameter can be obtained by using the integration formula involving \( \hat{a}, E(a_i) \), integration function. The mean value can serve as a point estimate for the unknown parameter. Alternative Bayesian estimators are the maximum a posterior (MAP) parameters. The uncertainty of the estimated parameters can be described by the variances and confidence intervals. To compute the confidence interval for the parameter \( \hat{a}_i \), where the confidence probability \( \text{Beta}(i) \) and the lower bound and upper bound of the interval are \((\text{low}_i, \text{up}_i)\) are used.

#### 2.2.3.2 Filtering, Adjustment and Validation for MEP

Information filtering means that the collected information needs to be checked before it is used for formulating the constraints as the basis of the MEP. Objective information (even without any subjective influence) Goldstein (2006) can be directly included. However, subjective information (such as suggestions from the experts) should be checked out. This objective information is collected proposed preparing a survey form filled by the experts with their suggestions. This form includes not only the suggestions but also confidence levels associated with the corresponding suggestions.
However, system cannot simply use the ranking of confidence $i_k$ levels to filter different experts’ opinions because some experts may be conservative, whereas some others may be aggressive or neutral. Therefore, the following method is suggested. First, the credible degree is defined as the probability of a certain expert’s suggestion being correct. Thus, approach should set up a threshold of credible degree, denoted by $c^*$. 

Suppose there are $K$ ranks. Then, at each rank, the credible degree of the ith expert (i.e., $i = 1, 2, \ldots, N$) can be calculated as

\[
C(i_k) = \frac{\text{No. of correct suggestions at rank } k \text{ in all prior projects of expert } i}{\text{Total no. of suggestions at rank } k \text{ in all prior projects of expert } i}
\]

Thus, regarding this expert i, those suggestions at the ranks of confidence levels should be filtered. Different experts have their own record, that is so, according to this criterion of credible Degree, the different experts’ opinions (whether conservative or aggressive) can be filtered, which is more fair and more reasonable than simply using an absolute rank to filter the suggestions. The threshold value $c^*$ could be initialized according to the specific requirement of the user, such as the minimum credibility that the user trusts. The minimum credibility here means that the threshold of credible degree can be increased after the adjustment steps.

The second step is to further check and adjust the information during the test on the current software. The newly observed data can be utilized as per this step. The prior distribution is not only a factor in the Bayesian formula but also a prediction of the current project software. If the prior distribution is much different from the real tendency of the newly
observed data, it means that the prior distribution might be inappropriate for
the current project/software. Therefore, adjustment should be carried out
according to this criterion.

The posterior distributions validate the posterior of modules by
using the following method:

- Calculate the mean values of the posterior distributions and
  obtain parameters.
- The parameters obtained can be applied to the test.
- Compare the predicted TimeToFailures with the real observed
  TimeToFailures, by calculating the mean square error.

\[ M(t) = b(1 - \exp(-a(t))), \quad a > 0 \]  \hspace{1cm} (2.4)

- If the mean square error is \( M(t) > d \) (Threshold value) then the
  model does not fit the observed data. The adjustment (such as
  trying another model or further filtering the subjective
  information) should be applied.

### 2.2.3.3 Adjustment for MEP

The details are elaborated in the following steps:

**Step 1**: There is a prior distribution \( p(x) \) with respect to a certain
parameter, given a set of newly observed data. Also, set up a
threshold for adjustment, which is a Probability \( p \).

**Step 2**: With the newly observed data, estimate the parameter (such as
using MLE) \( x ' \).
Step 3: Derive $x'/2$ by solving integration formulae.

Step 4: If $x' < x/2$ or $x' > x1$, the adjustment should be triggered. It means that the estimated parameter $x_0$ is located at the two extreme tails of the prior distribution, which is out of the middle interval with the probability

Step 5: Do the adjustment (such as increasing the credibility degree $c^*$ for information filtering and then recalculating the prior distribution with the newly filtered information via MEP) and then repeat Step 1.

The worst case is that $c^*$ finally increases to 1, which means that all information has to be filtered. Under this condition, the above steps should be terminated and then switched to the non informative prior. The value of ‘p’ can be set up in accordance with the user’s requirement of credibility degree such as the probability of the estimated parameter not being at either tail of the prior distribution

2.2.3.4 Information validation

Finally, when the test finishes using Musa’s (1975) and University Management System’s data sets, the posterior distributions can be derived based on MEP and BA. Then, validate the posterior distribution /model by using the following method: The mean values of the posterior distributions can be used for deriving the parameters. Then, these parameters can be applied to predict the TTFs during the test. Compare the predicted TTFs with the real observed TTFs, such as by calculating the mean square error. If the mean square error is too large over a certain preset threshold, it means that the model does not fit the observed data. Then, the adjustment (such as trying
another model or further filtering the subjective information) should be applied.

Table 2.1 Dataset by Musa (1975)

<table>
<thead>
<tr>
<th>Time(week) and Failure Interval Length</th>
<th>Time(week) and Failure Interval Length</th>
<th>Time(week) and Failure Interval Length</th>
<th>Time(week) and Failure Interval Length</th>
<th>Time(week) and Failure Interval Length</th>
<th>Time(week) and Failure Interval Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3</td>
<td>21 325</td>
<td>41 97</td>
<td>61 0.1</td>
<td>81 445</td>
<td>101 30</td>
</tr>
<tr>
<td>2 30</td>
<td>22 55</td>
<td>42 263</td>
<td>62 232</td>
<td>82 296</td>
<td>102 143</td>
</tr>
<tr>
<td>3 113</td>
<td>23 242</td>
<td>43 452</td>
<td>63 330</td>
<td>83 1755</td>
<td>103 108</td>
</tr>
<tr>
<td>4 81</td>
<td>24 68</td>
<td>44 255</td>
<td>64 365</td>
<td>84 1064</td>
<td>104 0.1</td>
</tr>
<tr>
<td>5 115</td>
<td>25 422</td>
<td>45 197</td>
<td>65 1222</td>
<td>85 1783</td>
<td>105 3110</td>
</tr>
<tr>
<td>6 9</td>
<td>26 180</td>
<td>46 193</td>
<td>66 543</td>
<td>86 860</td>
<td>106 1247</td>
</tr>
<tr>
<td>7 2</td>
<td>27 10</td>
<td>47 6</td>
<td>67 10</td>
<td>87 983</td>
<td>107 943</td>
</tr>
<tr>
<td>8 91</td>
<td>28 1146</td>
<td>48 79</td>
<td>68 16</td>
<td>88 707</td>
<td>108 700</td>
</tr>
<tr>
<td>9 112</td>
<td>29 600</td>
<td>49 816</td>
<td>69 529</td>
<td>89 33</td>
<td>109 875</td>
</tr>
<tr>
<td>10 15</td>
<td>30 15</td>
<td>50 1351</td>
<td>70 379</td>
<td>90 868</td>
<td>110 245</td>
</tr>
<tr>
<td>11 138</td>
<td>31 36</td>
<td>51 148</td>
<td>71 44</td>
<td>91 724</td>
<td>111 729</td>
</tr>
<tr>
<td>12 50</td>
<td>32 4</td>
<td>52 21</td>
<td>72 129</td>
<td>92 2323</td>
<td>112 1897</td>
</tr>
<tr>
<td>13 77</td>
<td>33 0.1</td>
<td>53 233</td>
<td>73 810</td>
<td>93 2930</td>
<td>113 447</td>
</tr>
<tr>
<td>14 24</td>
<td>34 8</td>
<td>54 134</td>
<td>74 290</td>
<td>94 1461</td>
<td>114 386</td>
</tr>
<tr>
<td>15 108</td>
<td>35 227</td>
<td>55 357</td>
<td>75 300</td>
<td>95 843</td>
<td>115 446</td>
</tr>
<tr>
<td>16 88</td>
<td>36 65</td>
<td>56 193</td>
<td>76 529</td>
<td>96 12</td>
<td>116 122</td>
</tr>
<tr>
<td>17 670</td>
<td>37 176</td>
<td>57 236</td>
<td>77 281</td>
<td>97 261</td>
<td>117 990</td>
</tr>
<tr>
<td>18 120</td>
<td>38 58</td>
<td>58 31</td>
<td>78 160</td>
<td>98 1800</td>
<td>118 948</td>
</tr>
<tr>
<td>19 26</td>
<td>39 457</td>
<td>59 369</td>
<td>79 828</td>
<td>99 865</td>
<td>119 1082</td>
</tr>
<tr>
<td>20 114</td>
<td>40 300</td>
<td>60 748</td>
<td>80 1011</td>
<td>100 1435</td>
<td>120 22</td>
</tr>
</tbody>
</table>
### Table 2.2 Data Set of University Management System

<table>
<thead>
<tr>
<th>Time(week) and Failure Interval Length</th>
<th>Time(week) and Failure Interval Length</th>
<th>Time(week) and Failure Interval Length</th>
<th>Time(week) and Failure Interval Length</th>
<th>Time(week) and Failure Interval Length</th>
<th>Time(week) and Failure Interval Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  60</td>
<td>21  104</td>
<td>41  9</td>
<td>61  89</td>
<td>81  126</td>
<td>101  57</td>
</tr>
<tr>
<td>2  30</td>
<td>22  16</td>
<td>42  4</td>
<td>62  85</td>
<td>82  36</td>
<td>102  1</td>
</tr>
<tr>
<td>3  540</td>
<td>23  9</td>
<td>43  66</td>
<td>63  54</td>
<td>83  54</td>
<td>103  2</td>
</tr>
<tr>
<td>4  67</td>
<td>24  10</td>
<td>44  0.5</td>
<td>64  3</td>
<td>84  15</td>
<td>104  5</td>
</tr>
<tr>
<td>5  40</td>
<td>25  12</td>
<td>45  18</td>
<td>65  15</td>
<td>85  16</td>
<td>105  1</td>
</tr>
<tr>
<td>6  23</td>
<td>26  4</td>
<td>46  15</td>
<td>66  6</td>
<td>86  154</td>
<td>106  17</td>
</tr>
<tr>
<td>7  5</td>
<td>27  10</td>
<td>47  75</td>
<td>67  8</td>
<td>87  84</td>
<td>107  14</td>
</tr>
<tr>
<td>8  53</td>
<td>28  82</td>
<td>48  30</td>
<td>68  36</td>
<td>88  92</td>
<td>108  2</td>
</tr>
<tr>
<td>9  4</td>
<td>29  6</td>
<td>49  116</td>
<td>69  98</td>
<td>89  247</td>
<td>109  87</td>
</tr>
<tr>
<td>10 16</td>
<td>30  54</td>
<td>50  14</td>
<td>70  32</td>
<td>90  244</td>
<td>110  19</td>
</tr>
<tr>
<td>11 94</td>
<td>31  25</td>
<td>51  15</td>
<td>71  36</td>
<td>91  60</td>
<td>111  29</td>
</tr>
<tr>
<td>12 15</td>
<td>32  43</td>
<td>52  41</td>
<td>72  5</td>
<td>92  5</td>
<td>112  0.5</td>
</tr>
<tr>
<td>13 5</td>
<td>33  12</td>
<td>53  1</td>
<td>73  9</td>
<td>93  2</td>
<td>113  61</td>
</tr>
<tr>
<td>14 90</td>
<td>34  48</td>
<td>54  99</td>
<td>74  60</td>
<td>94  5</td>
<td>114  118</td>
</tr>
<tr>
<td>15 77</td>
<td>35  23</td>
<td>55  9</td>
<td>75  34</td>
<td>95  130</td>
<td>115  20</td>
</tr>
<tr>
<td>16 68</td>
<td>36  6</td>
<td>56  45</td>
<td>76  16</td>
<td>96  0.5</td>
<td>116  3</td>
</tr>
<tr>
<td>17 15</td>
<td>37  10</td>
<td>57  68</td>
<td>77  164</td>
<td>97  233</td>
<td>117  11</td>
</tr>
<tr>
<td>18 137</td>
<td>38  12</td>
<td>58  36</td>
<td>78  123</td>
<td>98  50</td>
<td>118  87</td>
</tr>
<tr>
<td>19 23</td>
<td>39  14</td>
<td>59  50</td>
<td>79  19</td>
<td>99  54</td>
<td>119  5</td>
</tr>
<tr>
<td>20 1</td>
<td>40  33</td>
<td>60  81</td>
<td>80  19</td>
<td>100  52</td>
<td>120  249</td>
</tr>
</tbody>
</table>
Table 2.3. Model of Reliability values

<table>
<thead>
<tr>
<th>K</th>
<th>L</th>
<th>( P(r_{\text{IL+t}}^2) )</th>
<th>m(t)</th>
<th>Rank</th>
<th>Reliability Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.981309</td>
<td>0</td>
<td>0</td>
<td>662.0769</td>
<td>242.4507</td>
<td>0.073452884</td>
</tr>
<tr>
<td>23.55142</td>
<td>0</td>
<td>0</td>
<td>391.2273</td>
<td>245.9143</td>
<td>0.936789997</td>
</tr>
<tr>
<td>1.358736</td>
<td>0</td>
<td>0</td>
<td>358.625</td>
<td>296.7931</td>
<td>0.911564803</td>
</tr>
<tr>
<td>1.59917</td>
<td>12.5</td>
<td>12.23</td>
<td>172.14</td>
<td>344.28</td>
<td>0.911564992</td>
</tr>
<tr>
<td>3.140189</td>
<td>0.5</td>
<td>0.345</td>
<td>344.28</td>
<td>325.84</td>
<td>0.595401972</td>
</tr>
<tr>
<td>6.476639</td>
<td>0.0739</td>
<td>0.07123</td>
<td>662.13</td>
<td>358.63</td>
<td>0.067763432</td>
</tr>
<tr>
<td>2.564065</td>
<td>0.198217</td>
<td>0.178345</td>
<td>637.55</td>
<td>345.23</td>
<td>0.014567221</td>
</tr>
<tr>
<td>6.476639</td>
<td>2.422</td>
<td>2.323</td>
<td>172.14</td>
<td>358.625</td>
<td>0.902433272</td>
</tr>
<tr>
<td>72.61686</td>
<td>15.68</td>
<td>15.23</td>
<td>344.283</td>
<td>342.24</td>
<td>0.665728529</td>
</tr>
<tr>
<td>24.94034</td>
<td>1.300728</td>
<td>1.300256</td>
<td>555.2903</td>
<td>358.34</td>
<td>0.825561533</td>
</tr>
</tbody>
</table>

2.2.4 Performance Evaluation

In the Figure 2.3, the uncertainty is common due to the unexpected and dynamic nature of the system. The parameters in the above analytical model for large-scale system reliability are also dynamic, for instance, communication speed and processing speed are not constant, but varied under different workloads or mutable bandwidths. The original model is assumed that the parameters of communication speed and processing speed are constant, which is just an approximation to reality. Standard Deviation value first increases and then stabilizes to average value which means there is no deviation thereafter.
Based Uncertainty Analysis

Figure 2.3  Graph between the Standard Deviation for the model with uncertain parameters of Speed.

Figure 2.4 Reliability prediction and comparative analysis
In general, the reliability of the system decreases as time increases but the reliability of posterior mean always lies above the mean of MLE. In the Figure 2.4 both point estimate methods of MLE and posterior mean can predict a close reliability trend, so the new method using the posterior mean can be an alternative way for the point estimate of the parameters. More importantly, using the posterior probability distribution, it can further analyze the uncertainty of the predicted reliability. In addition, the confidence intervals depend on both the prior distribution and the new observations.

From the analytical results of uncertainty analysis in Figure 2.5, it can been seen that, during the initial period of the reliability prediction, the system reliability is high, indicating that the uncertainty of the software reliability is low. Then, the confidence interval increases and reaches the maximum around the middle part. At the latter part, the mean value of system reliability is also small, so that the comparative uncertainty is still large.

**Figure 2.5 System analysis**
2.3 SOFTWARE RELIABILITY MODELS BASED ON LOGISTIC EFFORT TEST FUNCTION

The Logistic testing-effort function is practically acceptable and for helpful for modelling software reliability growth and providing a reasonable description of resource consumption. The present work integrates the logistic testing-effort function into S-shaped model. During the software-testing phase, significant test-effort is required, such as the number of test cases, human power, and CPU time. The test-effort during the testing phase and the time-dependent behaviour of development effort in the software development process can be described by a Weibull-type consumption curve Hung, C. Y., Lin, C. (2004). This model uses a logistic testing-effort function instead of the Weibull-type testing-effort consumption function Hung, C. Y., Lin, C. (2004) as the testing-effort function to describe the test effort patterns during the software development process.

This kind of testing-effort function differs from the Weibull-type function Hung and Lin (2004). It is a dynamic model, since the resource consumption is estimated from a set of variables that are interdependent. Basic assumptions are

a) The fault removal process follows the Non-Homogeneous Poisson Process (NHPP).

b) The software system is subject to failures at random times caused by faults remaining in the system.

c) The mean number of faults detected in the time interval (t, t+Δt) by the current test effort is proportional to the mean number of remaining faults in the system.
d) The proportionality is constant over time.

e) The consumption curve of testing effort is modeled by a logistic testing-effort function.

f) Each time a failure occurs, the fault that caused it, is immediately removed, and no new faults are introduced.

2.3.1 Proposed Architecture

The architecture of the proposed system is depicted in the Figure 2.6. The description of improving software reliability is achieved through integrated logistic test effort reliability model. During the software development process, failures occur randomly. Failure process analysis is done using NHPP model. Failure data is analyzed and given as input to this model. The outcome of this model is an improved reliability factor.

![Diagram of Proposed Architecture](image-url)

**Figure 2.6 Logistic –Exponential Function with Reliability Models**
2.3.1.1 **Enrich Yamda S-Shaped Model with Logistic Effort function**

The software reliability of this model is different from NHPP while considering error isolation. The initial phase testers are familiar with type of errors and residual faults become more difficult to uncover. Figure 2.7 gives the flowchart of Enrich Yamda S-Shaped Model.

**(a) Yamada shaped Model**

Yamada S shaped model is so called because the plotted curve is obtained in the form of S shape. The Yamada S-shaped model was originally proposed by Yamada and was a simple modification of the NHPP to obtain an S-shaped growth curve for the cumulative number of failures detected. The failure rate initially increases, reflecting the immaturity of the software testing process to identify and remove defects. Subsequently, when the process matures, the failure rate peaks and then constants. The basic assumptions of this model are:

a) All detected faults are either independent or dependent

b) The total number of faults is finite.

c) The detected dependent fault may not be removed immediately and it lags the fault detection process by t.

d) Introduction of new independent faults during debugging process.
(b) \textbf{Logistic Effort Function}

Since actual testing-effort data expresses various expenditure patterns, sometimes the testing-effort expenditures are difficult to be described by only a Exponential or Rayleigh curve. Although the Weibull-type curve Lin and Huang (2008) can fit the data well under the general software development environment, it will have an apparent peak phenomenon when the shape parameter $m>3$. Therefore, a logistic testing-effort function is used, which was first presented by Parr instead of the Weibull type testing-effort consumption function Hung and Lin (2004) as the testing-effort function to describe the test effort patterns during the software development process Su and Huang (2007). This obtained function differs from the Weibull-type function Hung and Lin (2004) described in the above subsection and was used to derive the form of the resource consumption curve of a project over its life cycle. The logistic testing-effort function has the following form:

The cumulative testing effort consumption in time $(0, t)$ is

$$w(t) = N/(1+Ae^{-at}) \quad (2.5)$$

and the current testing effort consumption

$$w(t) = dw(t)/dt = aANe^{-at}/(1+Ae^{-at})^2 \quad (2.6)$$

where $N$ is the total amount of testing effort to be eventually consumed, $a$ is the consumption rate of testing-effort expenditures, and $A$ is a constant. Therefore, we can see that $w(t)$ is a smooth bell-shaped function. The testing effort $w(t)$ reaches its maximum value at time $t_{max}=1/\alpha \ln A$. Figure 2.7 shows the flowchart of Enrich Yamda S-Shaped Model.
Figure 2.7 Flow Graph of Enrich Yamada S Shaped model
The mathematical function of logistic testing function is based on Yamada S shaped model as follows:

- **Mean value function**

\[ m(t) = a(1 - e^{-rW(t) - W(0)}) = a(1 - e^{-rW(t)}) \]  \hspace{1cm} (2.7)

- **Logistic testing effort function**

\[ w(t) = N/1 + Ae^{-r(t)} \]  \hspace{1cm} (2.8)

- **Testing effort reaches its maximum value**

\[ t_{max} = \ln A/\alpha(t) \]  \hspace{1cm} (2.9)

where \( m(t) \) = the expected mean number of faults detected in time \((0, t)\), \( w(t) \) = current testing-effort consumption at time \( t \), \( a \) = the expected number of initial faults, and \( r \) = error detection rate per unit testing-effort at testing time \( t \) that satisfies \( r > 0 \).

### 2.3.1.2 Frontal Rayleigh Model

The Frontal Rayleigh model is a special case of the Weibull distribution Hung and Lin (2004), which has been widely used for various reliability studies in various fields. Figure 2.8 shows the flow chart of Frontal Compared to the phase-based defect removal model. It is a formal parametric model that can be used for projecting the latent software defects when the development work is complete and the product is ready for shipment. In addition, this model provides an excellent framework for quality management.
Figure 2.8 Flow Graph of Frontal Rayleigh Model
(a) **Scientific Notation**

- \( f_0 \) = Expected number of initial faults.
- \( \rho \) = Proportion of independent faults.
- \( t \) = time in (days).
- \( b \) = Independent fault introduction rate while removing/fixing a detected fault.
- \( r \) = Fault detection rate of independent faults.
- \( \Omega \) = Fault detection rate of dependent faults.
- \( m_i(t) \) = MVF of the expected number of independent faults detected in time \((0, t)\).
- \( m_d(t) \) = MVF of the expected number of dependent faults detected in time \((0, t)\).

(b) **Finding Faults**

Finding total detected faults in failure software is the first step.

The total detected faults in time \((0, t)\) are given by

\[
m(t) = m_i(t) + m_d(t) \quad (2.10)
\]

There are two types of software faults available:
• Independent Faults ($m_i(t)$)

The rate of independent faults detected is proportional to the remaining faults.

$$m_i(t) = \left(\frac{f_0}{1-b}\right) \left[ 1 - (1 + r(1-b)t)e^{-(1-b)t} \right]$$  \hspace{1cm} (2.11)

• Dependent Faults ($m_d(t)$)

The rate of dependent faults detected is proportional to the remaining dependent faults in the system and to the ratio of independent faults removed at time ‘t’ to the total number of faults.

$$m_d(t) = (1-p)f_0 \left\{ -e^{(A+B)[r(1-b)t]} \right\}$$  \hspace{1cm} (2.12)

(c) Removing Faults

This model describes a S shaped curve for the cumulative number of faults detected, such that failure rate initially increases, and later decays.

$$A = r(1-b)t + 2e^{-(r(1-b)t)}$$  \hspace{1cm} (2.13)

$$B = r(1-b)te^{-(r(1-b)t)} - 2$$  \hspace{1cm} (2.14)

(d) Reliability Estimation

Removing all detected faults will presumably increase the reliability of the software. The software reliability defined as the probability is that a software failure does not occur in the time $(t, \Omega t)$ interval,

$$R(\Omega t/t) = e^{-(m(t+\Omega t)-m(t))} \hspace{1cm} t > 0, \hspace{0.2cm} \Omega t > 0$$  \hspace{1cm} (2.15)
Using this formula, reliability of the software is estimated. Reliability is one of the main factors to improve the quality of the software. Reliability value is calculated for both perfect debugging and imperfect debugging. Consider the value \( b = 0 \) for perfect debugging and \( b = 0.1 \) for imperfect debugging. The same value is considered for reliability estimation also. The calculated reliability values are displayed in the table 2.4

2.4 PERFORMANCE ANALYSIS

The model has interesting characteristics of a failure rate that does not monotonically decrease with time as in the case with most software reliability growth models.

Table 2.4 Estimated Software Reliability Values on Perfect and Imperfect Debugging

<table>
<thead>
<tr>
<th>Time(t) (days)</th>
<th>No of failures under perfect and imperfect debugging ( m(t) )</th>
<th>Software Reliability under perfect and imperfect debugging ( R(10/t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( b = 0 )</td>
<td>( b = 0 )</td>
</tr>
<tr>
<td>20</td>
<td>99.74463</td>
<td>118.03086</td>
</tr>
<tr>
<td>30</td>
<td>149.48137</td>
<td>173.90707</td>
</tr>
<tr>
<td>40</td>
<td>176.24162</td>
<td>204.742</td>
</tr>
<tr>
<td>50</td>
<td>192.40932</td>
<td>220.81943</td>
</tr>
<tr>
<td>60</td>
<td>202.3388</td>
<td>230.26341</td>
</tr>
<tr>
<td>70</td>
<td>208.94038</td>
<td>233.91383</td>
</tr>
<tr>
<td>80</td>
<td>212.99417</td>
<td>239.30876</td>
</tr>
<tr>
<td>90</td>
<td>213.36184</td>
<td>241.3321</td>
</tr>
<tr>
<td>100</td>
<td>217.18843</td>
<td>242.38233</td>
</tr>
</tbody>
</table>
Figure 2.9 Number of Failures under perfect debugging and Imperfect Debugging

Figure 2.10 Software Reliability Estimation under Perfect Debugging and Imperfect Debugging
In the Figure 2.10, the lines on the Graph show variation of software reliability with respect to testing time. Under imperfect debugging, the probability of execution of failure free software is 90%. Whereas under perfect debugging, the probability is 84%. This shows that the prediction of software reliability is more realistic and generalized and prediction of when to stop testing based on reliability of software is also possible.

2.5 CONCLUSION

This chapter proposed three approaches for improving quality through software reliability model. The first approach presented improved bayesian approach for uncertainty analysis. This proposed work gives the solution for uncertainty problems in reliability modeling on system level. This safe growth model solves the challenges for the dearth of data by using quality factors. The model gives expert knowledge, historical data, and developmental environments. This expert knowledge is involved in analyzing the uncertainty and for compensating insufficient failure data. After analyzing the problem, this work is further extended to more complicated systems that contain numerous components, each with its own respective distributions and uncertain parameters.

The second approach presented software reliability models based on logistic testing effort function to the Yamada ‘S’ Shaped model. The reliability of the software can be improved by predicting more no of errors when compared to the normal Yamada ‘S’ shaped model. Logistic testing effort function based Yamada ‘S’ shaped model predicted more no of failures
than any other model as evident from the table. Hence Yamada ‘S’ shaped model with logistic testing effort function can be applied to time and failure sensitive projects and generic software.