On Statistical Reliability Modeling:
Methods of Inference and Prediction

Lutfiah Ismael Alturk

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Department of Computing
Faculty of Engineering and Physical Sciences
University of Surrey

September 2007

© Lutfiah Alturk 2007
Acknowledgments

This Ph.D. thesis has been completed after four years of scientific research by cooperation between King Abdul Aziz University and the University of Surrey. It was very nice to work in a project where theory and practice merged so well.

In the first place, I would like to express my deepest gratitude and appreciation to Prof. Paul J. Krause, Professor of software engineering, for guiding me on this journey from the beginning until the end, for always being ready to answer my questions, for his helpful suggestions and discussions, and for his continuous encouragements and for reviewing my work in a constructive way.

Further, I would like also to extend my sincere appreciation to Dr. Samia A. Adham, Assistance Professor of Mathematical Statistics, for her support and continuous encouragement during the development of this work.

I am also grateful to the financial support I received from the Joint Supervision Program during the planning of this thesis.

I am very much indebted to my dearest husband, my parents and children, thank you for being so patient, loving and understanding and supporting.

Finally, many thanks go to every one who supported and assisted me in any way.
Abstract

Over the last 35 years many statistical models have been proposed for the quantitative evaluation of software reliability. All the existing reliability models show a wide variability in predictive validity across data sets.

In this thesis, we discuss several Non-Homogenous Poisson Process (NHPP) models comprehensively. A theoretical review of these models is provided. For the two types of data, count and time data, the likelihood equations are obtained. We then generalize a very popular existing Software Reliability Growth Model (SRGM), the Littlewood model. The mathematical expressions of some important software reliability measures for the resulting general model and also the general Weibull model are derived. This theoretical analysis enabled us to develop easily configurable software tools to perform empirical studies of a number of SRGMs. For these we used two of published data sets. We used three techniques to analyze the predictive validity of the several special cases of the above two general models. The results of these evaluations emphasize the problem of the inexistence of one standard model that can be used accurately for all applications.

Two models refinement approaches, recalibration and model combination, were then explored. For these, we used two published data sets, and also included two additional data sets from recent large-scale development projects in the consumer electronics industry. Evaluations of predictive validity showed that when used individually; neither approach was universally effective across our data sets. However, applying recalibration, then model combination did provide significant improvements in predictive validity.

Finally, several prediction problems associated with using this conventional type of modeling and some corresponding solutions are summarized.
Abbreviations and Symbols

AIC  Akaike information criterion
PLR  Prequential Likelihood Ratio
cdf  Cumulative distribution function
pdf  Probability density function
FDR  Fault detection rate
CPU  Central process unit
GAM\( (x; \alpha) \)  Gamma distribution with one parameter \( \alpha \)
Par\( (x; \alpha) \)  Pareto distribution with one parameter \( \alpha \)
i.i.d.  Identically independently distributed
NHPP  Non-homogeneous Poisson processes
HPP   Homogeneous Poisson processes
SRGM  Software reliability growth model
MLE   Maximum likelihood estimate
MTTF  Median time to failures
\( \mu \) TTF  Mean time to failures
KS    Kolmogorov-Smirnov
SSE   Sum of square errors
SRET  Software reliability evaluation tool
SRE   Software reliability engineering
P\{ \}  Probability of the event \{ \}
E\{ \}  Expectation of the random variable or quantity \{ \}
LOC   Line of code
KLOC  LOC in thousands
ELC   Equally-Weighted Linear Combined
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLC</td>
<td>Median-oriented Linear Combined</td>
</tr>
<tr>
<td>ULC</td>
<td>Unequally-Weighted Linear Combined</td>
</tr>
<tr>
<td>DLC</td>
<td>Dynamically-Weighted Linear Combined</td>
</tr>
</tbody>
</table>
Contents

1 Software Reliability: Fundamental Definitions & Concepts 15
  1.1 Introduction 15
  1.2 The necessity for software reliability 17
  1.3 Software reliability engineering concepts and definitions 18
  1.4 Comparison between hardware and software reliability 27
  1.5 Outline of the thesis 28

2 Overview of Software Reliability 31
  2.1 Introduction 31
  2.2 Software reliability model classification scheme 31
  2.3 Overview of reliability modeling 33
    2.3.1 Software and system failures 33
    2.3.2 Foundational SRGMs 34
    2.3.3 Complexity and other influences on software reliability 38
    2.3.4 Neural Network based approaches 39
    2.3.5 Guidelines for the use of SRGMs 39
    2.3.6 Balancing cost and benefit for reliability engineering 40
    2.3.7 Causal modeling for reliability assessment 41
    2.3.8 New approaches to improving predictive accuracy 42
    2.3.9 Conclusions 44
3 Measures of Software Reliability Model Uncertainty

3.1 Introduction

3.2 Predictive validity analysis techniques

3.2.1 Kolmogorov-Smirnov (KS) test statistic

3.2.2 Akaike information criterion (AIC)

3.2.3 The sum of square errors (SSE)

3.2.4 Chi-square ($\chi^2$) test statistic

3.2.5 The Prequential Likelihood Ratio (PLR)

3.2.6 The u-plot

3.2.7 The y-plot

4 Statistical Inferences and Predictions in Several NHPP Models

4.1 Introduction

4.2 Non-Homogeneous Poisson Process (NHPP) Model

4.3 Software Reliability Growth Models (SRGMs)

4.3.1 Exponential (G&O) SRGM

4.3.2 Modified exponential SRGM

4.3.3 S-shaped SRGM

4.3.4 Inflection S-shaped SRGM

4.3.5 Testing-effort dependent SRGM

4.3.6 K-stage Erlangian (gamma) growth curve model

4.3.7 Hyperexponential SRGM

4.3.8 Littlewood SRGM

4.3.9 Duane Model

4.3.10 Testing domain SRGM

4.3.11 Logistic and Gompertz SRGM

4.4 Estimation of the Parameters of the SRGMs

4.4.1 Estimation of the SRGMs Parameters in case of Count Data

4.4.1.1 K-Stage Erlangian (gamma) model
4.4.1.2 Modified exponential model 88
4.4.1.3 Inflection S-shaped model 90
4.4.1.4 Testing-effort dependent model 92
4.4.1.5 Hyperexponential model 93
4.4.1.6 Littlewood NHPP model 95
4.4.1.7 Duane model 97
4.4.1.8 The Gompertz growth curve model 98
4.4.1.9 The logistic model 99

4.4.2 Estimation of the SRGMs Parameters in case of Time Data 104
4.4.2.1 Estimation of the parameters of the K-stage Erlangian model 104
4.4.2.2 Estimation of the parameters of the modified exponential model 106
4.4.2.3 Estimation of the parameters of the inflection S-shaped SRGM 107
4.4.2.4 Estimation of the testing-effort dependent SRGM parameters 108
4.4.2.5 Estimation of Littlewood NHPP model’s parameters 109
4.4.2.6 Estimation of Duane model’s parameters 111
4.4.2.7 Estimation of the Gompertz and logistic growth models’ parameters 112

4.5 Data analysis 116
4.5.1 Data tables 116
4.5.2 Criteria for model’s comparison 119
4.5.3 The models 119
4.5.4 Estimation of parameters 120
4.5.5 Graphical description 121
4.5.6 Choice between Software Reliability Models 133
4.5.7 Conclusion 139

5 On a General Formulation of the Littlewood Model 142
5.1 Introduction 142
5.2 Theoretical and mathematical work that support the generalization 144
5.2.1 Jelinski and Moranda (JM) model
  5.2.1.1 Littlewood modification to the JM model
5.2.2 Rayleigh model
  5.2.2.1 Modification to the Rayleigh model
5.2.3 Weibull model
  5.2.3.1 Modification to the Weibull model

5.3 Software failure data and analysis
  5.3.1 Techniques for the analysis predictive accuracy
  5.3.2 The models
  5.3.3 Real data sets
  5.3.4 Parameter estimation
    5.3.4.1 Estimation the parameters of the Weibull model
    5.3.4.2 Estimation the parameters of the Burr type XII model
  5.3.5 Discussion and graphical description
  5.3.6 Conclusion and suggestions

6 On Using Refinement Approaches for SRGM Prediction Improvement

  6.1 Introduction
  6.2 Refinement techniques
    6.2.1 The Recalibration approach
    6.2.2 The Combination approach
  6.3 Application of the refinement techniques
    6.3.1 Data sets
    6.3.2 Conventional models used
    6.3.3 Techniques for the analysis predictive accuracy
    6.3.4 Discussion of results
      6.3.4.1 Analysis of Data 1
      6.3.4.2 Analysis of Data 2
      6.3.4.3 Analysis of Data 3
      6.3.4.4 Analysis of Data 4
6.3.5 Conclusions

7 Conclusions and suggestions for future research

7.1 Conclusions

7.2 Suggestions for future research

Appendix A. Proofs of some relations

A.1 Proof (1)

A.2 Proof (2)

Appendix B. References

The content of the CD that accompanies this thesis

Program 4.1

Program 5.1

Program 6.1
List of Tables

1.1 Comparison between software and hardware reliability 30
3.1 Examples of software quality/reliability assessment tools 56
4.1 Summary of the characteristics of different SRGMs 80
4.2 Likelihood equations for error-detection count data 101
4.3 Likelihood equations for failure-occurrence time data 114
4.4 NTDS data 117
4.5 Data on F11-D program 117
4.6 DACS data 118
4.7 Times to the events (DS4) 118
4.8 Summary of data analyses 120
4.9 The actual data and reliability prediction results for NTDS data 122
4.10 The actual data and reliability prediction results for data on F11-D program 124
4.11 The actual data and reliability prediction results for DACS data 125
4.12 The actual data and reliability prediction results for DS4 data 127
4.13 Summary of AIC values for some SRGMs 135
4.14 Summary of D values for some SRGMs at $\alpha = 0.05$ and $\alpha = 0.01$ 136
4.15 Summary of $\chi^2$ values for some SRGMs at $\alpha = 0.05$ 137
4.16 Summary of SSE values for some SRGMs 138
5.1 Summary of some characteristics of Weibull and Burr type XII models 154
5.2 Weibull model in cases of $b = 1, 2$, and $3$ respectively

5.3 Burr type XII model in cases of $\gamma = 1, 2$, and $3$ respectively

5.4 Summary of data analyses based on three special cases of Weibull model using two failure data sets

5.5 Summary of data analyses based on three special cases of Burr type XII model using two failure data sets

5.6 Predictive quality analysis results of two failure data sets using the PLR technique

5.7 Predictive quality analysis results of two failure data sets using the u-plot and the y-plot techniques

6.1 Philips failure data 1: execution times between successive failures in minutes; read from left to right, number of failures= 246

6.2 Philips failure data 2: execution times between successive failures in minutes; read from left to right, number of failures=312

6.3 Musa SS1A data: execution times between successive failures in seconds; read from left to right, number of failures=112

6.4 Musa SS1C data: execution times between successive failures in seconds; read from left to right, number of failures= 277

6.5 Predictive quality analysis of raw, recalibrated and combined predictions of four real data sets using the PLR technique

6.6 Predictive quality analysis of raw, recalibrated and combined predictions of four real data sets using the u-plot and the y-plot techniques
# List of Figures

1.1 Comparison of software failure rates and reliability with time 21  
1.2 Software reliability growth 22  
1.3 Data domain \((t_i, y_i), (i=1, 2, ..., n)\) 24  
1.4 Time domain data \(x_i, (i=1, 2, ...)\) 25  
3.1 Analysis and assessment procedures in the package SRET 55  
4.1 Random variables for software reliability measurement 59  
4.2 Data analysis based on some SRGMs 129  
4.3 Estimated expected number of remaining faults based on some SRGMs 131  
5.2 Plots of reliability functions based on some special cases of Weibull and Burr Type XII models using two failure data sets 163  
5.3 Plots of reliability based on three special cases of Weibull model using two failure data sets 164  
5.4 Plots of reliability based on three special cases of Burr Type XII model using two failure data sets 165  
5.5 Median plots based on some special cases of Weibull and Burr Type XII models using two failure data sets 166  
5.6 Failure rate plots of three special cases of Weibull model using two failure data sets 167  
5.7 Failure rate plots of \(L\) model using two failure data sets 168  
5.8 Failure rate plots of \(SB\) and \(THPB\) models using two failure data sets 169
5.9 The log (PLR) plots of raw predictions based on some special cases of Weibull and Burr Type XII models versus a reference model, the JM model using two failure data sets 173

5.10 Raw u-plots based on some special cases of Weibull and Burr Type XII models using two failure data sets 174

5.11 Raw y-plots based on some special cases of Weibull and Burr Type XII models using two failure data sets 175

6.1 Failures times in order of occurrence for several real data set 186

6.2 A refinement example using Data 1 of Table 6.1 193

6.3 A refinement example using Data 2 of Table 6.2 198

6.4 A refinement example using Data 3 of Table 6.3 203

6.5 A refinement example using Data 4 of Table 6.4 208
Chapter 1

Software Reliability: Fundamental Definitions & Concepts

1.1 Introduction

Software reliability is a relatively new concept, and although huge efforts have been made to standardize it, to date, no one definition of software reliability nor one method of measuring or predicting software reliability is universally accepted. Instead, there are many models and metrics available today for estimating software reliability and measuring characteristics of software. Some of these models work well for some applications, but to date no one model can be used in all situations. The question remaining today is whether or not it is possible for one model to be standardized for all applications, or whether a combination of models is appropriate. There is also a question of whether or not it is feasible to predict the reliability of software before it is even developed.

This thesis makes contributions to both the theoretical and the empirical aspects of software reliability assessment. Fault discovery is a random process, and so can be modeled as a Non-Homogenous Poisson Process (NHPP). Hence, we first conduct a very detailed study using various NHPP models. We begin our study by reviewing the essential concept of constructing a NHPP model. A theoretical review of these NHPP models is provided, and the mathematical formulas for some important reliability measures for all these models are also obtained. For the purpose of estimation, we find all the likelihood equations which can be applied to the two sorts of data, count data and time data. Four different model selection techniques are used to make a comparative application, in the case of time data, between
eleven NHPP models using four independent failure data sets. In addition graphical representations are given for this end. Very useful results are obtained from this application. In addition, our application combined with several other studies [see, e.g. Abdel-Ghaly et al. (1986), Khoshgoftaar and Woodcock (1991)] emphasize the impossibility of finding one standard prediction system that can work well in all projects.

By providing generalizations of some of the existing software reliability models, we can develop configurable software reliability estimation tools that better support practitioners. In our thesis, we also generalize one of the most common models, the Littlewood model. Theoretical and mathematical work that supports the generalization is given. The mathematical expressions of some important software reliability quantities for the proposed model and also the general case of the Jelinski-Moranda (JM) model are derived. Several software reliability models, as special cases of these two general models, are discussed by analyzing two sets of failure data. In this example we use the maximum likelihood method to estimate the parameters of all the underlying models. Based on these models, some useful quantitative measures for the software reliability assessment are represented graphically. Furthermore, we apply three techniques to analyze the predictive accuracy of the studied models. The resulting inaccurate prediction systems that are obtained from this example motivated us toward the enhancement approaches.

Then, we explore how these models can be successively refined to improve their predictive accuracy for specific software development projects. To evaluate our work on an extensive analysis of real-world data sets we use seven different models. We employ the maximum likelihood method for model parameter estimation. Two new development scenarios are used to try to enhance the inaccurate prediction systems. Three predictive analysis techniques are applied to examine the predictive performance of the underlying models before and after conducting the modification approaches. Finally, we end our thesis by discussing some prediction problems which are associated with using this traditional type of modeling and some useful solutions are also proposed.

This first chapter addresses the necessity for software reliability. Some important concepts and definitions of software reliability are described. These concepts provide the
basis for quantifying the reliability of a system. Differences between software and hardware reliability are also given. In the last section we will present the outline of the thesis explaining the objectives, the technical work and the results which are discussed in the next six chapters.

1.2 The necessity for software reliability

The proliferation of computers in everyday life has substantially increased our dependence on them. Software is becoming pervasive. Everything from insurance rates to hotel reservations to long distance telephone calls are performed using software intensive systems. More importantly, software systems are used in life-critical applications, such as aircraft flight control and hospital monitoring devices where a failure could be catastrophic and could result in loss of life. Software systems are also used in areas such as banking and the stock market where a failure could result in loss of opportunity, and possibly, loss of money.

In parallel with this steadily expanding usage, customers are now expecting more reliable software. Many government contracts are now requiring that an established level of software reliability be achieved. Software has also become part of the system reliability allocations on many government contracts. Commercial clients are also requiring more reliable systems, and many are attempting to establish the same criteria as the government for development of reliable software. Financial institutions, medical institutions, the government, communication corporations, and other corporations are in position of being legally liable for software that is not accurate, that causes inconvenience to end users, and that causes end users to lose profits. In addition to being liable, users and developers of software are also facing increasing maintenance costs.

The cost of developing software is increasing. Data from a variety of sources show that for many systems developing software is becoming one of the major costs of any complex system, if not the major cost. As a result, software reliability becomes a very important issue in the last decade and there is a greater need for it.
1.3 Software reliability engineering concepts and definitions

In this section we define some important terms from software reliability engineering terminology. First we begin with defining the computer program as a set of computer instructions that executes within some processor (or set of connected processors) and relates to the accomplishment of some major functions. More than one program can execute simultaneously on a single processor. Hence, a software system can be defined as a collection of computer programs, procedures, rules, and their associated documentation, and data.

The term software reliability engineering (SRE) was not invented until the late 1960s. At that time concerns about the software crisis with software being expensive, bug-ridden and impossible to maintain, led to the notion that a move towards greater discipline in software development processes could resolve the problem. Hence, software engineering was born. SRE is defined as the methodical approach to the development, operation, maintenance, and retirement of computer software. It provides a software engineer or manager the means to estimate and measure the rate of failure occurrence in software. The main focus of SRE is on how the customers will use the product in their environment. Essentially, the claim is that good SRE approaches can significantly enhance software reliability. By statistical method in SRE we mean a unified framework for quantifying uncertainty, for updating it in the light of data, and for making decisions in its presence [Singpurwalla and Wilson (1999)].

The quality and reliability of the software must be considered in the early design phases, and maintained throughout the software life cycle. A software development process lifecycle can be defined as the period of time in which the software is created, developed and used. Essentially, a software lifecycle consists of the following five successive activities:

1. Analysis: the most important activity and the foundation of creating a successful software product. The objective of this activity is to identify the requirements and provide specifications for the dependant activities.
2. Design: the concern of this activity is to provide a high-level of the software system that will execute as required. This activity consists of two levels: system architecture design and detailed design.
3. Coding: this activity includes translating the design into the code of a programming language, beginning when the design documents are baselined.

4. Testing: testing is the process of executing a program to locate faults. The main goal in this activity is to affirm the quality of the software system by detecting and removing software faults, to reveal the presence of all specified functionality in the software, to estimate the operational reliability of the software. A test with a high probability of finding an undetected error is considered a good test. The input space is the set of all possible input states that can occur during the operation of the program. Similarly, the output space is the set of all possible output states for a given software and input space. While the environment of a software is created by counting the possible input states and their probabilities of occurrence. It can be changed with time. Because of problems of dimensionality it is very difficult to test a large software system. Therefore, failure probabilities are typically inferred from testing a sample of all possible input states.

5. Operation: this is the final activity in the software life cycle. This includes activities such as installation, training, support, and maintenance [Pham (2000)].

The operational profile of a system is defined as the set of operations that the software can execute along with the probability with which they will occur. Determining the operational profile is an important part of test planning, which generally occurs substantially ahead of the system test stage proper. This can be a contentious topic. Some engineers may deny the possibility of predicting an operational profile in advance of release of a software product.

In the early phases of a software lifecycle, a predictive model is needed because no failure data are available. This type of model predicts the number of initial faults in the software before testing. Software debugging is the activity to isolate faults and eliminate underlying errors. In the testing phase, the software reliability ideally improves through perfect debugging; a fault is removed with certainty whenever a failure occurs, and as a result the number of remaining faults is a decreasing function of debugging time. With an imperfect debugging assumption, faults may or may not be removed, introduced, or changed at each debugging session, so the number of remaining faults may decrease or increase. After the
release of a software program, the addition of a new modules, removal of old ones, removal of detected errors, mixing of newly and previously written code, change of user environment, and change of hardware and management involvement all have to be considered in the evaluation of software reliability. An evolution model is thus needed.

During the software testing activities, the problem of assessing software reliability arises. The generally accepted definition of software reliability is the probability of failure-free operation of a computer program in a specified environment for a specified time [Musa and Okumoto (1983) and Lyu and Nikson (1992)]. Mathematically, the reliability $R(t)$ is the probability that a system will execute successfully in the interval from time 0 to time $t$:

$$R(t) = P(T > t) = 1 - F(t),$$  \hspace{1cm} (1.1)

where $T$ is a random variable denoting the failure time.

In contrast, unreliability $F(t)$, the failure distribution function, is defined as the probability that the system will fail by time $t$.

$$F(t) = \int_0^t f(t) \, dt,$$  \hspace{1cm} (1.2)

where $f(t)$ is the probability density function that describe our uncertainty about when the component will fail.

In software reliability engineering, reliability is typically illustrated by the failure intensity which is a measure of the frequency of system failures as seen by users. More formally, we define the failure intensity as:

$$f(t) = \frac{F(t)}{R(t)}$$  \hspace{1cm} (1.3)

The relation of the failure intensity to reliability is shown in Figure 1.1.
During the final testing phases, if we assume perfect debugging of the software, the failure rate decreases due to the discovery and the removal of the software faults and the software reliability tends to increase or grow. The final stage of the test phase is determining when a software product is ready for release. This phase of software development can be considered to be the acceptance phase [Kaufman et al. (1997)].

During testing, computer software is subject to failures caused by faults latent in the software. Test data such as the times of software failures or the number of detected faults can then be observed. If it is assumed that the correction of faults does not introduce any new faults, the cumulative number of detected faults increases as they are corrected, and the mean time interval between software failures becomes longer. These cases can be shown in Figures 1.2.a, 1.2.b, and 1.2.c. This means that the probability of no failure occurring in a fixed time-interval, that is the reliability, increases with the progress of software testing.
The software cannot be improved unless there are development techniques for doing so. This whole process cannot be optimized with respect to cost and time without management of the measurement, analysis, improvement, and development methodologies, and the procedures, resources, and schedules. *Measurements and analyses* are necessary in order to indicate to both management and development the state of the product and the process. The measurements and analyses techniques include:

- Software reliability metrics,
- software reliability models,
- and software analysis.
Software reliability metrics: which are measures of some aspect of the software product or process; itself can be divided into the following four classes:

- **Software metrics**: software size is an example of this class which reflects complexity, development effort, and reliability of software. Line of code (LOC), or LOC in thousands (KLOC), is an initial method to measuring software size. Complexity is believed to be a key influence on software reliability, so representing complexity is very important. Complexity-oriented metrics is a method of determining the complexity of a program's control structure, by simplifying the code into a graphical representation. Test coverage metrics are a way of estimating fault and reliability by performing tests on software systems, based on the assumption that the software reliability is a function of the portion of software that has been successfully tested.

- **Project management metrics**: better software systems can only be achieved by good management. The claim of exponents of the Capability Maturity Model and other process improvement initiatives is that higher reliability can be obtained by using better development process, and project cost increases when developers use inappropriate processes.

- **Process metrics**: based on the assumption that the quality of the software is a direct function of the process, process metrics can be used to assess, control, and improve the reliability of software.

- **Fault and failure metrics**: the objective of observing fault and failure metrics is to obtain failure free operation. Both the number of faults found during testing and the failures reported by users after release are collected, summarized, and analyzed to achieve this objective. The observed failure data is consequently used to calculate failure density, Mean time between failures or other parameters to quantify or predict software reliability.

Software reliability models: for the most part, model the failures occurring because of the software. There are many types of software metrics and reliability models. A software reliability growth model (SRGM) is defined as a mathematical relationship between the time spent in operating a software system and the software reliability measures such as the
cumulative number of detected faults and the time-interval between software failures [Ramamoorthy and Bastani (1982), Yamada (1989)]. Using the SRGMs, we can estimate several software reliability measures such as the initial fault content, the mean time between failures, the expected number of remaining faults, the software reliability function, and so on. During the last 25 years, several models have been proposed. Leading SRGMs will be mentioned in Chapter 2 as part of our literature review.

The testing of particular software is designed to activate certain branches in order to detect, via system failures, the faults which cause it. Let $N$ be the number of faults in a computer program and $T_i, i = 1, 2, \ldots, N$, be the random detection times of these faults. Most of the models in the literature pertaining to this matter are the so called "time domain" models which consider the time till a fault is activated as a random variable realized according to some stochastic process. The other types of models are called "data domain" models. These models consider the software system as a finite population of functional units. Figures 1.3 and 1.4 show the two types of data appropriate for these two types of models.

Figure 1.3: Data domain $(t_i, y_i), (i = 1, 2, \ldots, n)$
Observed Predicted
Failure number
$t = 0$
Cumulative failure time
Interfailure Time

Figure 1.4: Time domain data $x_i, (i = 1, 2, \ldots)$

Obviously the interval structure of any program will affect its reliability. It should be taken into account in a reliability model. The history of a typical program is shown in Figures 1.3 and 1.4. At time zero the program is executed and works satisfactorily until time $t_1$, when the first failure occurs. The programmer then repairs the program, it works satisfactorily for time $t_2$, is repaired again and so on, till it reaches a stage when it will run some significant time before failing. This means that compilation errors and execution errors have been largely eliminated, and modeling of the reliability growth at these stages of program writing will probably be a discrete time exercise.

In developing models for software reliability, particular attention must be given to the data that is available for analysis. One parameter that specifies the quality of the system is the \textit{mean time to failure (MTTF)}. This is the expected time when the next failure is observed due to software faults. This parameter is defined as:

$$
\mu \text{MTTF} = \int_0^\infty t \, f(t) \, dt .
$$

(1.4)

It is shown that the analysis of the $\mu \text{MTTF}$ provides the prerequisite information for reliability analysis [Johnson (1989)].
Some of the proposed metrics and models have been shown over time to be invalid. For example, in the 1970s the most commonly used metrics was errors per executable source line of code. Over time, however, it has been found that this metrics is not as valuable or valid as originally thought [Neufelder (1993)].

Software models and metrics are most useful when used in conjunction with each other and when used during the most appropriate phase or phases of the life cycle. It is not effective to use all existing metrics, nor it is effective to use the incorrect metric.

It is necessary that metrics and models be used and chosen discriminately. There are some metrics that should be used in every project, such as distribution of error type and total error counts. However, some metrics should only be used if there is reason to believe that the metrics will expose some valuable information that will improve the development process. Software reliability models should also be used discriminately because the assumptions of each model vary and may not fit the characteristics of a given software project. A common mistake when implementing some of the reliability models is to adjust the development environment to fit the model instead of finding the model that fits the development environment.

Software analyses: before the delivery of software, testing, verification, and validation are necessary steps. Several analysis methods such as trend analysis, fault-tree analysis, Orthogonal Defect classification and formal methods, etc, can also be used to enable development personnel to find errors in the software while the software is still in a laboratory environment, and therefore minimize the possibility of defect occurrence after release and therefore improve software reliability. Software measurement and analysis should not be an isolated part of the process. It must be integrated into the process to be successful.

After operation of a software intensive system, data can be collected and analyzed to study the behavior of software failures. Fault tolerance technique which is defined as systems capable of recovery from hardware or software failure to give uninterrupted real-time service is a way of handling unexpected failure and helping to minimize fault occurrence or impact of the fault on the system.
1.4 Comparison between hardware and software reliability

By definition, a software intensive system consists of two major components: hardware and software. System failures may be classified as software or hardware or other (such as user). Although extensive research has been carried out on hardware reliability, the growing importance of software dictates that the focus shift to software reliability. There are many differences between the reliability and testing concepts and techniques of hardware and software. Therefore, a comparison of software and hardware reliability would be useful in developing software reliability modeling. In fact, it is very important to understand exactly what software and hardware failures are before comparing software reliability to hardware reliability.

First, let us define software error, fault, and failure. By **software error** we mean the programmer or system analyst's cerebral mistake. A **software fault** is the manifestation of a software error in the code. Hence, a software fault is generated when an error is made. In applications, errors can be classified (typically) into four classes: critical, high, moderate, and low. A **software failure** is defined as the deviation of the software output, caused by a software fault, from what is expected according to the requirements. A failure is considered corrected when the fault causing it has been removed. There may be faults in the code that can be ignored since these faults will never manifest themselves as visible failure. In other words, some software faults may or may not cause software failures, but all the software failures are caused by software faults.

**Hardware failure**, on the other hand, is a hardware fault that causes a system failure. The mechanisms for hardware failures are not the same as for software failures. However both hardware and software have the effect of a degradation of the system. Hardware typically fails due physical stress, time, wearing out, the elements such as wind, rain, snow, and temperature, and other environmental factors.
Software failure as mentioned above is due to programmer or designer errors committed during requirement, design, code, test, or maintenance of the software. Software does not wear out or burn out in the same sense as hardware. The software itself does not fail, unless flaws within the software result in a failure in its dependent system. Table 1.1 summarizes the differences and similarities between software and hardware reliability.

In the next section the main contents of each chapter are surveyed.

1.5 Outline of the thesis

The thesis consists of 7 Chapters. Chapter 1 is a preliminary chapter divided into 4 main sections. In addition to this section the first one is an introductory section. Section 2 gives reasons for the importance of software reliability. Some definitions and concepts that help to build technical foundation in software reliability engineering and provide the basis for quantifying the reliability of a system are discussed in Section 3. In Section 4 we make a comparison between software and hardware reliability.

In Chapter 2 we review the literature on different aspects of software reliability engineering. Specifically, an overview on software reliability models and suggested solutions of problems arising when the models are applied for different purposes are presented.

Chapter 3 presents seven different goodness-of-fit criteria to compare between SRGMs. Applications using these criteria are given in Chapters 4, 5 and 6.

Chapter 4 reviews the main concept of constructing a Non-homogenous Poisson Process (NHPP) software model. Some important software reliability models for the failure phenomenon based on NHPPs and their characteristics such as error detection rate per fault, failure intensity, expected number of errors remaining in the software at testing time, software reliability function and the time between failures are considered. The likelihood equations which can be applied to the two sorts of data, count data and time data, are obtained for each model. Applications using four data sets and eleven of these models are conducted in the case of time data. The estimation of model parameters, and consequently their characteristics, using the maximum likelihood estimate, MLE, technique are obtained and the predicted
curves are presented graphically. Furthermore, four different model selection techniques to compare between SRGMs using the same four data sets on the eleven studied models are applied. Lastly, graphical representations for comparison of these models based on predictive accuracy, are given in this chapter.

In Chapter 5, we provide foundational theoretical and mathematical work that supports the generalization of one of the most popular software reliability models; the Littlewood model. Some special cases of the proposed model and another famous general reliability model, Weibull model, are considered. Data analysis results illustrating the potential problem of obtaining inaccurate prediction results when using this traditional type of modeling are presented. Some suggestions to improve measurement accuracy are given.

In Chapter 6 we present a methodical framework to conduct model evaluation of seven software reliability growth models using several techniques for the analysis of predictive accuracy. Two enhancement approaches which can greatly enhance the predictive quality of SRGMs are used.

The final chapter is concerned with the general conclusions of the research and suggestions for future research points that, as we hope, will be of interest to researchers in the field of software reliability.

In addition to these seven chapters, two appendices and a software CD are enclosed in the thesis. Appendix A contains proofs of two relations in Chapter 4. The references are collected in Appendix B. The software CD contains the code of three Java programs. These programs are used and their functions are explained in detail, in Chapters 4, 5, and 6.
<table>
<thead>
<tr>
<th>Software reliability</th>
<th>Hardware reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software failures are mainly caused by design faults, which are harder to visualize, classify, detect, and correct.</td>
<td>Hardware failures are mostly caused by physical faults, as a result hardware reliability is a much easier measure to obtain and analyze.</td>
</tr>
<tr>
<td>Software changes its behavior when its environment changes (obsolescence).</td>
<td>Hardware changes its behavior even in a constant environment (wear out).</td>
</tr>
<tr>
<td>Software failures only occur when the software is executing.</td>
<td>Hardware failures can be caused by material deterioration even though the system is not operated.</td>
</tr>
<tr>
<td>It is possible to have software that is bug-free and so will never experience failure for any mission time.</td>
<td>Hardware experiences deterioration with use and is thus prone to failure over time.</td>
</tr>
<tr>
<td>No standard components for software, except some standardized logic structures.</td>
<td>Hardware components can be standardized.</td>
</tr>
<tr>
<td>Software reliability can be improved by increasing the testing effort and by correcting observed faults. Redundancy can not improve software reliability.</td>
<td>Better design, better material, extensively-tested standards components, applying redundancy will help improve hardware reliability.</td>
</tr>
<tr>
<td>Software failures mostly happen without warning.</td>
<td>Warning usually occurs before hardware failure.</td>
</tr>
<tr>
<td>If we do not consider the program development the failure rate is statistically non-increasing.</td>
<td>The failure rate has a bathtub curve. The burn-in is similar to the software debugging state</td>
</tr>
<tr>
<td>The set of all possible inputs to the software is generally huge and so is the sequence in which the inputs are received by the software; that is, the operational profile of the software is not unique. Therefore software essentially requires infinite testing.</td>
<td>Hardware can usually be tested exhaustively.</td>
</tr>
</tbody>
</table>

Table I.1: Comparison between software and hardware reliability
Chapter 2

Overview of the Software Reliability

2.1 Introduction

Computer systems have come to be crucial tools in various activities of society. Lately, since more breakdowns of computer system are caused by software failures than by hardware ones, it is of great importance to produce reliable software systems by using software reliability engineering approaches. The quality assurance of software has become one of the most important problems in the development of software production technologies. Over the past few years software engineers have been besieged by a barrage of papers on different aspects of software reliability. Several approaches for improving and assessing computer software reliability have been presented. In this chapter a scheme for classifying software reliability models is given in Section 2.2. Section 2.3 provides some of the existing literature on software reliability models, model comparisons, stopping rules, and approaches to improve the software attribute of reliability.

2.2 Software reliability model classification scheme

One particular aspect of software reliability engineering (SRE) that has received the most attention is software reliability modeling. A classification scheme has been developed for software reliability models to permit relationships to be derived for groups of models, to highlight relationships among the models and suggest new models where gaps occur in the classification scheme. Using the classification scheme presented in [Musa and Okomoto (1983)] models are classified in the following currently used four model classes:
the exponential failure time class of models;
> the Weibull and the gamma failure time class of models;
> the infinite failure category model; and
> Bayesian models.

Each class of models can be described as follows:

1) Exponential failure time class of models: This class has the most papers in the literature on software reliability. Any finite failure model that requires that the functional form of the failure intensity function be exponential is an exponential failure time model. This type of model is delineated into two types [Musa and Okomoto (1983)]: binomial and Poisson. The binomial type in this class assumes a constant per-fault hazard rate. The Poisson, which can either be homogeneous (HPP) or non-homogeneous (NHPP) processes, also assumes constant per-fault hazard rate, and in addition, it assumes an exponential time to failure of an individual fault. The Jelinski–Moranda (JM) model is an early example of a binomial type model and provides the framework for later models. The NHPP model [Farr (1996)], the Schneidewind model [Schneidewind (1975, 1993)] and the Musa model [Musa et al. (1987)] are examples of the Poisson type model. The hyperexponential model [Ohba (1984b) and Laprie et al. (1991)] is an extension of this class of models.

2) Weibull and gamma failure time class of models: If the finite failure model requires that the functional form of the failure intensity function be either a Weibull or a gamma distribution, then the model belongs to this class. The advantage of the Weibull model over that of the exponential failure time model is that it can accommodate increasing, decreasing or constant failure rates.

3) Infinite failure time model: In this model class, it is assumed that software will never be completely fault free, that is

\[
\lim_{t \to \infty} H(t) = \infty ,
\]

(2.1)

where \( H(t) \) is the mean value function that presents the number of failures that have occurred from time 0 to time \( t \). One of the earliest infinite models is the Duane model [Duane (1964)].
4) Bayesian models: The Bayesian approach uses prior and posterior distributions to incorporate past and current data for reliability prediction. This type of analysis requires that the distribution of failure times be known a priori. The Littlewood–Verall (1973, 1974) model is an example of this model class.

All the models belonging to the above classes attempt to make predictions of software reliability in the later stages of the life cycle. In the next chapters we will present and further analyse some known models belonging to the above-mentioned classes.

2.3 Overview of Reliability Modeling

2.3.1 Software and system failures

Software reliability is now an established research area. Because of the rapid development of software technology and the increasing dependence on software systems in industrial and commercial companies, the study of software reliability has attracted the attention of both researchers and engineers.

Software is usually a part of a more complex system and software failures may only be one of many classes of failure. Laprie and Kanoun (1992) studied software reliability and availability modeling combining the hardware counterpart. In addition they considered the conventional reliability growth study and models of stochastic processes. System structure is also incorporated into the reliability and availability study.

Keene and Chris (1992) discussed in their paper the similarities and differences between software and hardware failures in the system reliability content. They showed that software failures occur more often than hardware failures in a system. Various concepts are discussed. Markov modeling is suggested for system modeling and analysis.

A mathematical model of software is called a reliability model if it is used to obtain a measure of the reliability of the software. There are many interesting reliability parameters, such as the number of faults in the software, the expected number of failures in a certain time
interval and the failure intensity in the field of application, but usually we are interested in the probability that software will fulfill its intended function without failure in a specified time.

2.3.2 Foundational SRGMs

Over the years, effort made to estimate and measure software reliability has led to the development of numerous models. There exist two approaches to tackling the problem of software reliability modeling. The first approach is known as the black-box approach e.g. [Littlewood and Verall (1974), Goel and Okumoto (1979)], where the program is regarded as a black box and thus the internal structure of the program is ignored. The second approach is to take into account the internal structure of the program. Relatively little work on the latter has been reported in the literature e.g. [Shooman (1976)].

Essentially, the work on software reliability models started in the 1970's. Today the number of existing models exceeds two hundred with more models being developed every year. It is important to have an overview of the most commonly used models for the purposes of model selection and decision-making. However, this is becoming more difficult because of the large number of models proposed.

In this section we will mention some important references on various types of SRGMs that appeared in the literature. The paper which is most cited in the software reliability literature is by Jelinski and Moranda (1972). Their model JM (the first black-box model), since its publication, has been widely discussed theoretically and practically. It is assumed that a number of faults exist in the software and through testing, debugging problems are found and faults are removed. Thus the reliability increases over time. Times between failures are assumed to be exponential with a parameter that is proportional to the number of remaining faults. The JM model has later been both modified and generalized by various authors. One modification of the JM model, by Moranda (1975b), allows unequal change in software failure intensity after debugging. The suggested model is called the geometric deeutrophication model and the original assumption that the failure intensity is proportional to the current fault content is replaced by a more realistic one. On this basis the Moranda model would seem to be more realistic and useful. Although the JM model has been widely used, it suffers from difficulties associated with parameter estimation (giving decreasing reliability).
Littlewood and Verrall (1981) presented a simple necessary and sufficient condition for the maximum likelihood estimates to be finite and suggested that this condition be tested prior to using the model.

Dickson et al. (1972) stated in their paper the basic definitions that are useful in software reliability analysis. A model for the residual faults is developed and other related quantities are derived.

Musa (1975) introduced a model that became known as the Musa execution time model. An important fact is that model parameters are related to the software development process and this appears to be attractive for many practical engineers. Musa and Okumoto (1984) introduced a model modifying the Musa model, based on the assumption that the failure rate decreases exponentially with the expected number of failures experienced. The assumption is more realistic since most of the existing models assume that all software faults have the same failure rate. Ohba (1984b) presented a detailed study of a few interesting models and indicated the use of several other models, such as Gompertz model and logistic curves as used in software reliability study in Japan.

Crow and Singpurwalla (1984) presented a model which can be used in the case where the times between failures cannot be assumed to be independent and exponentially distributed. This model is especially useful in identifying possible clustering of software failures and it can also be used to predict future failures.

The history of Bayesian software reliability models was initiated in the early 1970's. The Littlewood–Verrall (1973) model as originally proposed and studied is one of the leading software reliability models. It assumes that times between failures are exponentially distributed with a parameter that is treated as a random variable which is assumed to have a Gamma prior distribution. This model is relatively simple and analytical expressions are obtained. One of the few papers on Bayesian NHPP models is by Higgins and Tsokos (1981) and the underlying model is the Duane model. Jewell (1985) presented a study on Bayesian software reliability models. The basic model is the JM model and an extension is provided to another Bayesian model by Langberg and Singpurwalla (1985). The main assumption is that
the distribution of the unknown number of faults is assumed to be Poisson whose parameter has a Beta prior distribution.

In some models, it is assumed that the software, after some debugging, contains no further faults. Becker and Camarinopoulos (1990) presented a Bayesian model of the software failure rate. They introduced a simple class of conjugate prior distribution in order to get some mathematically tractable results. The model, which provides a reduction in failure rate due to the correction, can be seen as an interesting Bayesian extension of the existing ones.

A model is called a failure-counting model if it deals with the number of failures in a certain time interval. A representative type, which also forms the largest group, is the modeling of the failure process by a non-homogeneous Poisson process (NHPP); such models are commonly known as NHPP models. The observed number of failures as a function of time can be fitted to the mean value function and predictions of future failure counts can be made. The first paper presenting a software reliability model based on NHPP is by Schneidewind (1975). Many SRGMs used by practitioners, belong to this category as discussed in Bastani and Ramamoorthy (1986). Goel and Okumoto (1979) introduced a model which became known as the Goel–Okumoto (G & O) model. This model originates from Schneidewind model.

Xie and Zhao (1992) showed that several NHPP models can be derived based on the general assumption made by Schneidewind. They noted that Schneidewind proposed many approaches worth considering. A few of them, model both the software fault detection process and the software fault correction process. Yamada et al. (1983, 1984) have done further work in this area and they have developed several related models, among them, the S-shaped SRGMs are especially interesting and have been extensively used Kanoun et al. (1991). Xia and Kumar (1992) developed a model which incorporates the factor of learning process during the testing phase using all the assumptions of Goel and Okumoto, with the additional assumption that the error detection rate is proportional to the tester’s experience which is an increasing function of time $t$ (say $f(t)$), i.e. $0 \leq f(t) \leq 1$, $f'(t) > 0$ and $f''(t) \leq 0$. They also showed that the S-shaped models (delayed and Inflection) are particular cases of the new model. These models will be discussed, in detail in our thesis.
An infinite failures model is the Duane model. This was derived from the hardware reliability area. Duane (1964) observed several hardware applications in which the rate of failure occurrence was in a power law form in operating time. Crow (1974) added the assumption that this process was an NHPP. The Duane model can be applied in software reliability although some new problems may arise. Littlewood (1984) modified this model to overcome a problem associated with it, i.e. the problem of infinite failure rate at time zero.

In Kareer et al. (1990) two generalizations of the G & O model, namely the model of two types of faults and the S-shaped NHPP model, are combined. The software faults are classified into two groups according to their severity and two S-shaped NHPP models are used. Optimum release policies that minimize the cost subject to achieving a given reliability are also discussed based on the model assumptions.

Khoshgoftaar and Woodcock (1991) studied the K-stage Erlangian growth curve model, which is a NHPP one. Its mean value function has the same growth curve. By introducing a third parameter, this model nicely combines the well-known G & O model and the S-shaped model. Furthermore, the Akaike information criterion (AIC), which is based on a kind of log-likelihood function, is used for the selection of the best model. It is argued that this method is simple, concise and accurate. Studies on the comparison between models using available data have been conducted by many authors e.g. [Ohtera et al. (1991), Downs and Scott (1992)].

One of the recent models is the log-power (NHPP) model which is suitable for software reliability analysis in practice. This model has been suggested by Zhao and Xie (1992) who studied it and found that, when the model is valid, the plot of the cumulative number of failures versus the running time on a log-double-log scale, tends to be a straight line. The estimated parameters are the slope of the fitted line and its intercept on the vertical axis. The MLEs are also shown to be of closed analytical form and the model has good predictive accuracy for several of the published data sets.

Kremer (1983) considered a general class of software reliability models based on birth-death processes. Their model allows perfect removal, imperfect removal and incorrect
removal of detected software faults. The remaining number of faults after a debugging activity will be reduced by one, unchanged or increased by one according to a certain probability distribution. Many other models are in fact special cases of this type of model. In addition, several reliability measures were obtained.

A few models generalizing the existing ones are suggested by Ohba and Chou (1989) to cope with the case of imperfect debugging. After a review of conventional models, the definition of imperfect debugging is discussed. It is assumed that a detected fault is removed only with a certain probability. In this sense, the proposed models are improvements of conventional software reliability growth models. It is indicated that the exponential-type models are useful even if the assumption of perfect debugging is not valid. It is stressed that the interpretation of the model parameters, in general, has to be changed.

2.3.3 Complexity and other influences on software reliability

Most software complexity studies are related to the estimation of the number of faults in the software. Muson and Khoshgoftaar (1991) presented a unified approach using software metrics for the estimation of quantities related to software reliability and failure rate. The model is mainly a regression model, assuming a number of metrics are known. A new software metric, called the relative complexity metric, that combines a number of existing ones is introduced and how this can be used in reliability study is presented.

Fault tree analysis is a technique for studying the relationship between a system failure event and its causes. It has been used in reliability engineering for a long time. Stålthane (1989) was one of the earliest to demonstrate the use of the Fault tree technique in software reliability and simple examples were presented. This systems approach is useful, especially for modular software and software in safety-critical systems. In his paper, the use software fault tree analysis was presented in detail, together with the calculation of quantities such as criticality and diagnostic time.

Fault-tolerance is a technique for improving user-perceived reliability. Hence, reliability analysis is strongly related to fault-tolerance, although most of the published papers on fault-tolerance deal with fault-tolerance configuration rather than software reliability analysis. The
reliability achieved by various standard fault-tolerance techniques are compared in Scott and Gault (1987) and the relative cost of increasing software reliability is also discussed.

It is important to provide very high reliability in safety or mission critical applications. One way of handling unexpected failure is through the use of fault-tolerant computer systems, which are defined as systems capable of recovery from hardware or software failure to give uninterrupted real-time service. A considerable research in this area has been conducted over the last three decades [Abbott (1990), Belli and Jedrzejowicz (1990)].

2.3.4 Neural Network based approaches

Neural network models are highly idealized mathematical models that stem from the understanding of the biological nervous system. A number of attempts have been made at using them for software reliability analysis. Karunanithi et al. (1991) used a kind of feedforward neural network as the model for software reliability growth prediction in their study. A few data sets are used as a practical example and it is indicated that consistent behavior in prediction and a performance comparable to that of some other models can be obtained. In particular, it is observed that good consistent results are obtained using the neural network model.

2.3.5 Guidelines for the use of SRGMs

The existence of a number of different software reliability models means that the user of such models needs some guidelines for their application. The criteria that should be used in making comparisons have been the subject of considerable discussion among researchers. The criteria in approximate order of importance, as mentioned in Musa and Okumoto (1983), are:

(a) predictive validity,
(b) usefulness,
(c) quality of assumptions,
(d) applicability, and
(e) simplicity.
In 1985 Goel and others started describing processes in which each model could be tested to see how well it fits the data and predict future failure events. Abdel-Ghaly et al. (1986) compared the predictive quality of ten models using four different methods of comparison. Gray (1986) presented in his paper a framework for the modeling of software reliability. His opinion is that selecting a suitable model from the existing ones is more important than creating new models. He used advanced statistical theory to compare a range of parametric models. In (1991) Keiller and Miller presented a modeling approach, the use of the so-called “super models”, which is a model selection approach based on goodness of fit and quality-of-predictive criteria. In our thesis, there are three applications involving comparison between some software reliability models based in several failure data sets. All the model selection techniques that we are going to use in our applications will be explained in some detail in the next Chapter.

2.3.6 Balancing cost and benefit for reliability engineering

In current years, the costs of developing software have entailed a significant percentage of the total expense in a system development. Therefore, it is important to decide when to stop testing, or when to release the software to the market so that the total system cost is minimized, subject to the desired reliability level and other constraints.

The relationship between reliability and cost has always been important for software developers. As Zahedi and Ashrafi (1991) clearly state: “One can hardly perceive reliability without considering the cost of achieving it”. Moreover, the high cost of software development and maintenance is an important reason for the emphasis on producing reliable software [Reiss (1979)]. Since software developers operate within limited resources (time and money), there must be a tradeoff between reliability and cost. Software reliability and cost are often seen as two competing issues, with an action that leads to improvement in one being detrimental to the other. Ashrafi and Berman (1992) presented two optimization models for decision support tools for selecting available programs in the market. Information on reliability and cost of the available programs are considered as basic criteria for the selection. The objective is to maximize the average reliability of the software package, considering the tradeoff between reliability and cost of the programs.
Software testing is one of the important components in software development. An important question in the debugging process is, when to stop. The choice is usually based on one of the two criteria:

1. when the reliability has reached a given threshold,
2. when any further gain in reliability cannot be justified by the testing cost.

To use criterion 1, we need to predict the remaining (future) failure rate of the software if testing is stopped. Additionally, to use criterion 2, we need to identify a monetary cost equivalent for failures encountered by the user.

In (1977) Forman and Singpurwalla suggested a procedure for estimating the parameters of the reliability model and proposed a stopping rule for debugging the software. Later the same authors (1979) discussed an empirical stopping rule for debugging and testing computer software. They presented results on choosing a time interval for testing the hypothesis that a software system contains no errors, given certain cost and risk constraints.

Various stopping rules and software reliability models are composed by their ability to deal with the above two criteria. Among researchers who studied this topic are Ross (1985), Singpurwalla (1991), Dalal and Mallows (1992) and Ehrlich et al. (1993). Yang and Chao (1995) presented two new stopping rules, initiated by theoretical studies of the optimal stopping rules based on cost, which are more stable than other rules for a large variety of bug structures. The first stopping rules based on the basic execution and logarithmic Poisson models due to Musa et al. (1987), as well as the stopping rule by Dalal and Mallows (1990), work well for software with many relatively small bugs (with very low occurrence rate).

### 2.3.7 Causal modeling for reliability assessment

The use of Bayesian networks is a relatively recent approach that can be used to make predictions of numbers of software faults. Fenton and Neil (1999) provide a critical review of some defect prediction studies. They determine a number of theoretical and practical problems associated with those studies. They then proposed a model for software defect
prediction using Bayesian belief networks as an alternative approach which could help partly to solve the problems they identified.

Fenton et al. (2002) show that building predictive models can effectively support process control and risk management, rather than simple regression models that fail to take into account all the major casual influences on a project's quality goals.

For many reasons, these graphical probabilistic models are a promising candidate for assessing and controlling the quality of software products. The reasons, as mentioned in Krause et al. (2003), are as follows:

- Casual influences between variables in a particular domain can be easily modeled.
- In the absence of the empirical data in those areas of a problem domain, the Bayesian approach enables statistical inference to be augmented by expert judgment.
- According to the above, it is possible to add variables that correspond to process as well as product attributes in a software reliability model.
- Assigning probabilities to reliability predictions means that they can support sound decision making approaches using classical decision theory.

### 2.3.8 New approaches to improving predictive accuracy

Although SRGMs have been very intensely studied and investigated, there are still many problems that have been responsible for their potential inaccurate predictions in practice. A new approach for constructing SRGMs based on NHPPs has been proposed by some researchers. The main focus is to provide a method for software reliability modeling, which consider factors that may enhance the prediction capability of the software reliability models. Kuo et al. (2001) incorporated testing effort function and time-variable fault detection rates, FDR, into this type of modeling. Instead of the traditional assumption of constant FDR, they assumed that the FDR has three possible trends as time progresses: increasing, decreasing, or constant. To interpret these possible trends, they treated the FDR as a function of time. Then, they joined the testing effort function and time-variable FDR into a combined analysis for software reliability modeling. They showed that the combined model gives a more accurate prediction, and delineates the real failure data more faithfully.
Essentially, most of the existing models assume that all faults have equal probability of being detected during the software testing process, and the rate remains constant over the intervals between fault occurrences. In reality, the fault detection rate strongly depends on the skills of the testing team, program size and software testability. Thus, it may not be smooth and can be changed at some time moment called a change point. In (2005) Huang consolidated both generalized logistic testing-effort function and change point parameter into software reliability modeling. He showed that the proposed model has better predictive validity based on real failures experienced and gives a more accurate prediction.

However, most SRGMs assume that detected faults are immediately corrected. Actually, this assumption may not be realistic in practice. Specifically, all detected faults can be categorized as leading faults and dependent faults. Moreover, the fault correction process can be modeled as a delayed fault-detection process and it lags the detection process by a time dependent delay. Huang et al. (2006) proposed a SRGM based on NHPP by applying the ideas of fault dependency and time dependent delay. The presented results indicate that the proposed framework to incorporate both failure dependency and time-dependent delay functions for SRGM has a fairly accurate predictive capability.

Lyu and Nikora (1991) advocated that finding techniques to reuse the existing SRGMs may give better prediction capability than presenting a new SRGM. They considered a model called the equally-weighted linear combination model. Several predictive validity analysis techniques are used to assess the performance of this model. It is indicated, through detailed analysis using various data sets, that the presented model performs better than many others. In our thesis we will also form several linear combination models using either raw or recalibrated models. The resulted combined models are going to be included in a comparative application using some real failure data sets.

Instead of proposing new software reliability models Brocklehurst et al. (1990) suggested to enhance the existing ones. They mentioned in their paper that no model can be universally the best model, but most of the existing models can be improved. They introduced a recalibration approach to improve the software reliability prediction. They also analyzed
existing and simulated data using this approach and indicated that the relationship between the predicted and the true reliability can be estimated using the so-called U-plot technique, which is an approach to improve the existing models. In (1992) Brocklehurst and Littlewood described in their paper the U-plot technique and prequential likelihood ratio. These techniques help users not only select a good model, but also modify a model so that a better accuracy is achieved. In Chapter 6 we will try to improve the predictive validity of some conventional SRGMs by using the recalibration approach.

2.3.9 Conclusions

Because of the rapid growth of complexity in software systems, delivering reliable software products on time becomes a critical issue. Over the past thirty five years, many SRGMs have been suggested and studied for estimation of reliability growth of software products and making decisions during the software development process. Generally, SRGMs are applicable to the last stages of testing in software development. The predictability of those SRGMs has been investigated by comparing their predictions with sets of failure data. Although, SRGMs can provide very useful information about how to improve the reliability of software products, none of those SRGMs can capture a necessary amount of the software characteristics. There is no single model that is universal to all cases [Abdel-Ghaly et al. (1986), Khoshgoftaar and Woodcock (1991)]. The reason behind this is possibly that the developers design their models with assumptions that are good estimates of reality just in some situations. Many common assumptions made in these models cannot be strictly adhered to in practical applications and, of course, this limited the applicability and the effectiveness of those SRGMs. Moreover, the development techniques that aimed to refine the performance of those SRGMs give good results sometimes but failed to guarantee giving improved software reliability predictions all the times [Brocklehurst et al. (1990), Lyu and Nikora (1991)]. Eventually, Software reliability modeling is an interesting but complicated research area. By basing the modeling approach on a deep knowledge and understanding of the software development process, we hope that this approach will continue improve to help in establishing a software reliability theory.
Chapter 3

Measures of Software Reliability Model Uncertainty

3.1 Introduction

Early work in the field of software reliability centered around proposing new models [Ramamoorthy and Bastani (1982)]. Several models have been proposed, such as those mentioned in Chapter 2. Each model was shown to work well with a unique data set, but no model appeared to do well in all cases.

In (1985) Goel and others started describing a process in which each model could be tested to see how well it fits the data and predicts the future events. The assertion was that different models predict well only on certain data sets, and that by comparing the predictive quality of different models it is possible to select the best one for a given application. But how are different models to be compared?

Abdel-Ghaly et al. (1986) compared the predictive validity of ten models using four different methods of comparison. They showed that different methods of model selection resulted in different models being chosen. Also, some of their methods were rather subjective as to which model was better than another. So, a simple and objective selection method is needed [Khoshgoftaar and Woodcock (1992)]. From this point of view, selecting a suitable model from a large class of plausible models is an important problem in statistics. There is a
significant literature on model comparison e.g. [Lyu and Nikora (1991), Poore et al. (1993)]. Furthermore most papers presenting new models include some results on model comparison. In this Chapter we will study seven model selection techniques to evaluate the predictive validity of different SRGMs.

### 3.2 Predictive validity analysis techniques

The conventional software reliability models we investigate in this thesis depend on the successive inter-failure times, $T_1, T_2, ..., T_n$ which are observed during software testing, as the only input. Actually, our goal of using this type of modeling is, at any stage $i$, where $i=j, j+1, ..., n$, where $j$ is a number sufficiently large for the first prediction and $n$ is the total number of software failures observed from the selected software, to estimate the current and future reliability. After that, the performance of these models needs to be examined. As mentioned in the previous chapter, the criteria that should be used in making comparisons have been the subject of considerable discussion among researchers. For quick reference, the criteria in approximate order of importance, as mentioned in [Musa and Okumoto (1983)], are predictive validity, usefulness, quality of assumptions, applicability, and simplicity. In this thesis we only concern with analyzing the predictive validity, the first criterion and indeed the primary concern, of the models we investigate. To this end, we will use the following model selection techniques:

a) Kolmogorov-Smirnov (KS) test statistic.
b) Akaike information criterion (AIC).
c) Chi-square ($\chi^2$) test statistic.
d) The sum of square errors (SSE).
e) The u-plot.
f) The y-plot.
g) The Prequential Likelihood Ratio (PLR)

We will now provide brief descriptions of each of these criteria.
3.2.1 Kolmogorov-Smirnov (KS) test statistic

Goodness-of-fit based on the KS test statistic [Neave and Worthington (1982)] for a NHPP model [Yamada (1989) and Goel (1982)], which is useful even if the sample size of the observed data is small, can be illustrated as follows:

Let \( \hat{H}(t_i) \) and \( \hat{H}(t_n) \) be the estimated expected cumulative number of faults detected up to testing time \( t_i \) and \( t_n \), respectively. These will be defined in detail in Chapter 4. In addition, let \( y_i \) be the cumulative number of detected faults, in a given time \( (0, t_i) \), \( (i = 1, 2, ..., n; 0 \leq t_0 \leq t_1 \leq ... \leq t_n) \) as shown in Figure 1.3. Now assume that

\[
D_i = \max \left\{ \left| \frac{\hat{H}(t_i) - y_i}{y_i} \right|, \left| \frac{\hat{H}(t_n) - y_n}{y_n} \right|, \left| \frac{\hat{H}(t_n) - y_{i-1}}{y_n} \right| \right\},
\]

and

\[
D = \max\{D_i\}
\]

is the KS test statistic in the case of the error-detection count data. Essentially, it is calculating the maximum vertical distance between the actual and estimated cumulative numbers of faults.

For the failure occurrence time data \( s_i \), \( (i = 1, 2, ..., n) \), which is shown in Figure 1.4, the KS test statistic, that we are going to use in our application in Chapter 4, is given by

\[
D = \max D_i
\]

where

\[
D_i = \max \left\{ \left| \frac{\hat{H}(s_i) - i}{n-1} \right|, \left| \frac{\hat{H}(s_i) - i-1}{n-1} \right| \right\},
\]

The values of the test statistics in equations (3.2) and (3.3) are compared with the critical values for the KS statistics \( D_{n,\alpha} \) and \( D_{n-1,\alpha} \) with the sample sizes \( n \) and \( n-1 \) for specified level of significance \( \alpha \), respectively. If the calculated value \( D \) in equation (3.2) or (3.3) is less
than the selected critical value, then it can be concluded that the observed data fits the applied SRGM with the specified level of confidence.

### 3.2.2 Akaike Information Criterion (AIC)

Before engaging in the construction of a model, we must accept that there are no actual models. Certainly, models only approximate reality. The question then is to find which model would best estimate reality given the real data we have observed. In other words, we are attempting to minimize the loss of information. Kullback and Leibler (1951) addressed such issues and developed a measure, the Kullback-Leibler information measure to represent the information lost when approximating reality. In (1974), Akaike established a relationship between the maximum likelihood, which is an estimation method used in many statistical analyses, and the Kullback-Leibler information. He developed an information criterion to estimate the Kullback-Leibler information, the Akaike information criterion (AIC).

The AIC resulted from relating the entropy principle from statistical mechanics to information theory. Akaike began with classical hypothesis testing using the log likelihood function. From this relation he was able to determine the amount of information in an observation. He then found that part of this corresponds to the entropy measurement in statistical mechanics. For more details on the issue, see [Khoshgoftaar and Woodcock (1992)].

Suppose that \( f \) and \( g \) are the probability density functions of the true and the hypothetical distributions, respectively. Define

\[
B(f;g) = - \int f(z) \ln \left( \frac{f(z)}{g(z)} \right) dz,
\]

Note that \( B(f;g) = 0 \) if and only if \( f = g \), and that \( B(f;g) \leq 0 \). Akaike has argued that the Kullback-Leibler (1951) measure, \(-B(f;g)\), may be used as a measure of discrepancy between \( f \) and \( g \).
Suppose that the data set X of N observations is given. Akaike suggests that the purpose of statistical analysis of X is the prediction of some future observation Y. The predictive distribution of Y as a function of the available data X is \( g(Y|X) \), the predictive distribution of Y as a function of the available data X. If the true distribution of Y is given by \( f(y) \), then the goodness of \( g(Y|X) \) as an estimate of \( f(y) \) is measured by the entropy of \( f(y) \) with respect to \( g(Y|X) \) as:

\[
B[f(y); g(Y|X)] = - \int f(y) \ln \left( \frac{f(y)}{g(Y|X)} \right) \, dy \\
= \int f(y) \ln g(Y|X) \, dy - \int f(y) \ln f(y) \, dy \\
= E_Y \left[ \ln g(Y|X) \right] - c , \quad (3.7)
\]

where \( E_Y \) denotes the expectation with respect to the distribution of Y and \( c \) is a constant.

The goodness of the estimation procedure specified by \( g(Y|X) \) is measured by:

\[
E_x E_Y \left( \ln g(Y|X) \right) . \quad (3.8)
\]

Suppose that X and Y are independent. If

\[
g(y|X) = g(y|\theta) ,
\]

i.e. a distribution specified by a fixed parameter vector \( \theta \), then

\[
\ln g(X|X) = \ln g(X|\theta) , \quad (3.9)
\]

which is exactly the classical definition of the log likelihood of the model specified by \( g(\theta) \), conventionally called the log likelihood of the parameter \( \theta \). Therefore, from equations (3.8) and (3.9) we have

\[
E_x \ln g(X|\theta) = E_x E_Y \ln g(Y|X) . \quad (3.10)
\]

Referring to equation (3.7), Akaike has proposed that the model which maximizes
\[ \ln g(X|\theta) - c \]

should be adopted. This is equivalent to the problem of minimizing

\[ -2 \ln g(X|\theta) + 2c, \quad (3.11) \]

that is

\[ \text{AIC} = -2 \left( \text{log likelihood function at its maximum likelihood estimator} \right) + 2 \left( \text{number of parameters fitted when maximizing the likelihood function} \right) \]

The value of the AIC for a single data set has no meaning. It becomes interesting when it is compared to the AIC of a series of models specified a priori; the model with the lowest AIC value indicates the best model for certain data. The models can then be ranked from best to worse (i.e., low to high AIC values). The AIC not only measures how well the data fits the model, but also how well the model predicts the future failure behavior of the software system. The AIC provides an objective way of determining which model among a set of models is best, as we do not rely on a specified level of significance \( \alpha \). It is rigorous, founded on solid statistical principles (i.e., maximum likelihood), yet easy to calculate and interpret.

### 3.2.3 The sum of square errors (SSE)

This is perhaps the most well known of all the fitness tests. A residual is the difference between an observed and a predicted value of a function. The sum of squares error, SSE, is the sum of squares of the residuals. The SSE can be used to compare the estimation accuracy of the applied models and can be defined as follow:

\[ \text{SSE} = \sum_{i=1}^{n} \left( H(s_i) - \hat{H}(s_i) \right)^2, \quad (3.12) \]
where $H(s_i)$ is the cumulative number of faults and $\hat{H}(s_i)$ is its estimate, ($i = 1, 2, \ldots, n$). Using this test, the "best" model for the data is the model with the lowest SSE value.

### 3.2.4 Chi-square ($\chi^2$) test statistic

If we wish to provide a quantitative comparison between models, we can use the Chi-square ($\chi^2$) test statistic. We can obtain the values of the $\chi^2$ statistic for the NHPP models by:

$$
\chi^2 = \sum_{i=1}^{n} \frac{(H(s_i) - \hat{H}(s_i))^2}{\hat{H}(s_i)}.
$$

We then compare the calculated values of $\chi^2$ in equation (3.13) for software reliability models that are fitted to real failure data sets, with the critical values in a $\chi^2$-Table. For a sample size of $n-1$ and a specified level of significance $\alpha$, if the calculated value is less than the critical value $\chi^2_{n-1, \alpha}$ selected from the table, it can be conclude that the real data set fits the applied SRGM (to the specified level of significance).

### 3.2.5 The Prequential Likelihood Ratio (PLR)

The Prequential Likelihood Ratio (PLR) is intended as a global comparison of goodness for one prediction system versus another. Suppose we have two prediction systems, A and B. A comparison of these two prediction systems over a range of predictions of $T_j, T_{j+1}, \ldots, T_{j+n}$ can be defined as:

$$
\text{PLR}_{AB} = \prod_{k=j}^{n} \frac{\hat{f}_k^A(t_k)}{\hat{f}_k^B(t_k)},
$$

where $\hat{f}_k^A(t_k)$ and $\hat{f}_k^B(t_k)$ are the two predictive densities for the prediction system A and B respectively. To explain how this technique works we assume that the prediction system A is
more accurate than B. This means if we compare the two predictive densities at $t_k$ we will find $\hat{f}_k^A(t_k)$ is closer to the true pdf $f_k(t_k)$. The observation $t_k$ will tend to take a value where $f_k(t_k)$ and $\hat{f}_k^A(t_k)$ is large, rather than in the tail where $\hat{f}_k^B(t_k)$ is not so large. Thus to choose the model A as being the better of the two models, $\frac{\hat{f}_k^A(t_k)}{\hat{f}_k^B(t_k)}$ will tend to be larger than 1 and this should tend to increase with $i$ if the prediction system A is more accurate than B.

Actually, even if the model A is consistently more accurate than B, there is no guarantee that a single $\frac{\hat{f}_k^A(t_k)}{\hat{f}_k^B(t_k)}$ will always be greater than one. But we can expect the plot of PLR or for convenience, its log, to exhibit an overall increase with some fluctuation. We will use this technique in this thesis to compare the accuracy of some of the software reliability growth models. For doing this we will select one of the studied models as a reference and conduct pairwise comparisons of all of them against it, as we mentioned above.

### 3.2.6 The u-Plot

The PLR technique can not help us to estimate the relationship between the predicted and the true reliability and therefore does not allow us to determine which of the models is objectively accurate. Littlewood (1980) proposed what is now a widely accepted technique for evaluating the predictive quality of a software reliability model; the u-plot. This technique allows a user to estimate the relationship between the predicted and the true reliability. It is well known that if the random variable $T_i$ truly had the distribution $\hat{F}_i(t_i)$ (i.e. if the prediction and the truth were identical) then the random variable $U_i = \frac{\hat{F}_i(t_i)}{F_i(t_i)}$ would be uniformly distributed on $(0,1)$. If we were to observe the realization $t_i$ of $T_i$, and calculate $u_i = \frac{\hat{F}_i(t_i)}{F_i(t_i)}$ then the number $u_i$ will be a realization of a uniform random variable. When we do this for a sequence of predictions, then we should get a sequence $\{u_i\}$, which looks like a random sample from a uniform distribution. Any departure from the uniformity will indicate some kind of deviation between the sequence of predictions $\hat{F}_i(t_i)$ and the truth $F_i(t_i)$. We will use Kolmogorov-Smirnov (KS) distance, the maximum vertical deviation of the plot
from the line of unit slope, to test the significant of the departures in order to judge the performance of existing models.

### 3.2.7 The y-Plot

This technique is very important to ensure that the errors in prediction (which maybe present in the u-plot) are at least consistent, so that the u-plot prediction result can be trusted. As we mentioned above the \( u_i \) sequence should look like a sequence of independent, identically distributed uniform random variables on \((0, 1)\). Since the range, \((0, 1)\), remains constant, any trend will be difficult to detect in the \( u_i \) sequence, which will tend to look very regular. If, however, we make the transformation \( x_i = -\log(1 - u_i) \), we produce a sequence of numbers that should look like realizations of independent, identically distributed unit exponential random variables. That is, the sequence should look like the realization of the successive inter-event times of a homogeneous Poisson process; any trend in the \( u_i \)'s will show itself as a non-constant rate for this process. One way to test the trend in a Poisson process is by normalizing the whole transformed sequence onto \((0,1)\). That is, for \( i = j, j+1, ..., n \), we define

\[
y_i = \frac{\sum_{k=j}^{i} x_k}{\sum_{k=j}^{n} x_k}
\]

As with the u-plot we can construct the y-plot by drawing a step function with steps of size \( 1/(n-j+2) \) at the points \( y_j, y_{j+1}, ..., y_n \) from the left on the interval \((0, 1)\). In the u-plot the order of occurrence of the u's disappears while the y-plot preserves this ordering. If a prediction result has captured the trend of reliability growth in the data then the y's should be identically distributed and the y-plot should show no significant departure from the line of unit slope.

### 3.3 Software Reliability Tools

Yamada et al. (1989) developed a software reliability evaluation tool, SRET, in which the analysis and assessment procedures shown in Figure 3.1 are implemented in a program
package, using the BASIC language. SRET uses three SRGMs based on a NHPP, which are the exponential, delayed S-shaped, and inflection S-shaped SRGMs, and two deterministic ones, which are the logistic and Gompertz growth curve models. SRET can be very useful in the testing phase, since the software managers can perform software reliability assessment easily in an interactive mode without knowing the details of the process of data analysis.

Several tools based on similar ideas have been developed. Table 3.1 shows examples of tools developed by some Japanese computer manufacturers and software houses.

We will be using all the above-mentioned techniques during the empirical evaluations in the later part of this thesis. In order to be able to customise the analysis techniques to our own needs and integrate them effectively with our SRGM implementations, we have developed a suite of Java programs for evaluating the predictive accuracy. The programs have been tested using the Java™ 2 SDK, Standard Edition, version 1.3.1, on a personal computer using the XP operating system, but will execute on any other machine that has the appropriate version of a Java Virtual Machine installed.
Collect Data as Error Counts or Failure Times

Choose an Appropriate SRGM

Estimate Model Parameters

Analyze Failure-Occurrence Time Data

$S_k (k = 1, 2, ..., n)$

Choose Another Model

Perform Goodness-of-Fit Test

Accept

Obtain Reliability Assessment Measures

Undetected Errors $n(t)\$

Software Reliability $R(\chi|\chi)$

Others [e.g. MTBF(t)]

Decision Making

- Estimate Time for Release
- Predict Additional Testing Time to Reach Goal
- Etc.

Figure 3.1: Analysis and assessment procedures in the package SRET.
<table>
<thead>
<tr>
<th>Tool</th>
<th>Integrated SRGM</th>
<th>Developer</th>
<th>Reference</th>
</tr>
</thead>
</table>
| SORPS                | • exponential SRGM  
                    • delayed S-shaped SRGM  
                    • inflection S-shaped SRGM | IBM Japan     | Ohba (1984)   |
| SPARC                | • delayed S-shaped SRGM  
                    • logistic growth curve model  
                    • Gompertz growth curve model | Toshiba       | Yamada (1991) |
| Software Reliability | • exponential SRGM  
                    • delayed S-shaped SRGM  
                    • inflection S-shaped SRGM  
                    • logistic growth curve model  
                    • Gompertz growth curve model | Toshiba Engineering | Komuro (1987) |
| SOREM                | • exponential SRGM  
                    • delayed S-shaped SRGM  
                    • logistic growth curve model  
                    • Gompertz growth curve model | NEC           | Uemura (1990)  |

Table 3.1: Examples of software quality / reliability assessment Tools
Chapter 4

Statistical Inferences and Predictions in Several NHPP Models

4.1 Introduction

As we have discussed in the introduction, measuring and improving the quality of computer software has, over the last few years, been receiving a great deal of attention. The following concepts are known as the standard characteristics of software quality: reliability, functionality, usability, efficiency, maintainability and portability. Software reliability is an important quality characteristic as a taken-for-granted quality [Matsumoto and Ohno (1989)]. It is defined as the probability of no occurrence of a software failure during a certain period on a specified condition. Early work in the field of software reliability is centered on proposing new reliability assessment and prediction models [Ohba (1984a)]. Over the last thirty-five years a significant number of models have been proposed, and each has been shown to work well on certain data sets. However, as yet it has not been possible to develop a single preferred model that provides a "best" fit to all data sets.

A possible way forward is to identify a process by which each model could be tested to see how well it fits a given data set, and the model that best fits the data used for reliability prediction. Abdel-Ghaly et al. (1986) compared the predictive quality of ten models using the following four different methods of comparison:
• the u-plot,
• the y-plot,
• measure of noise and
• prequential likelihood.

They showed that different methods of model selection techniques resulted in different SRGMs being chosen. Also, some of their methods were rather subjective as to which model was better than another. Khoshgoftaar and Woodcock [1992] proposed the Akaike Information Criterion (AIC) as an objective selection method. They claimed that this will always select the “best” method. The AIC criterion takes into account both goodness of fit and predictive power of a model given a specific data set. So there are grounds for preferring this as a selection method. However we show here that other valid statistical measures of a goodness of fit could be chosen, and each will provide a different ranking of candidate models for a given data set (and these rankings will change across different data sets).

We argue that it may be misguided to try and select a single model to try and predict reliability from a given set of software test data. All the software reliability models make certain assumptions about the distribution of faults within a software product, and the nature of the fault discovery process (software testing). Instead of trying to identify a single model, the use of several models can provide information both about

• the validity of those assumptions, and
• the confidence with which we can make a reliability estimate.

The aim of this Chapter is to discuss several SRGMs. They are mostly based on NHPPs that describe the time-dependent behavior of software failures occurring during the testing phase. Firstly, in Section 4.2 we will review the essential concept of constructing an NHPP software model. In Section 4.3, we will provide a theoretical foundation of several NHPP models, and the characteristics of these models will also be obtained. In section 4.4, for the purpose of parameter estimation, the likelihood equations which can be applied to the two sorts of data, count and time data, are acquired for all the discussed models. In Section 4.5 we
conduct an application to compare the preference rankings of eleven NHPP models against four independent data sets. The estimation of model parameters, and consequently their characteristics, using the maximum likelihood estimate (MLE) technique are discussed. The predicted curves are presented graphically. Four candidate statistical tests of fitness are used to compare the underlying models. We can use the goodness-of-fit techniques to test the models' assumptions, and perhaps reject certain models. We argue that it may then be more useful to consider the distribution of reliability predictions over the remaining models, rather than identify a preferred single value. Finally, we end our analysis with some conclusions that will lead into the next phase of our research programme.

4.2 Non-homogeneous Poisson process (NHPP) model

The following assumptions are usually made in the area of software reliability growth modeling:

1) Software is subject to failures at random times caused by faults latent in the software.
2) A software failure is caused by a software fault.
3) Each time a failure occurs, the fault that caused it is immediately removed, and no new errors are introduced.

For software reliability measurement, define the following random variables:

\{J(t) = i\}

Figure 4.1: Random variables for software reliability measurement
\( X_i = \) the time-interval between the \((i-1)^{th}\) and \(i^{th}\) failures.

\( S_i = \) the time to the \(i^{th}\) failure.

\[
S_i = \sum_{k=1}^{i} X_k, \quad X_i = S_i - S_{i-1}, \quad (i = 1, 2, \ldots; X_0 = S_0 = 0).
\] (4.1)

Suppose \( j(t) \), the cumulative number of software faults detected by the time \( t \), is so large that it can be treated as a continuous function of \( t \). Assume that the number of undetected faults at any time is finite, hence \( j(t) \) is a bounded, non-decreasing function of \( t \). Further assume that the total number of errors to be eventually detected \( j(\infty) = a \). Then, we have

\[
j(t) = \begin{cases} 
0 & \text{when } t = 0 \\
\quad a & \text{when } t = \infty
\end{cases}
\] (4.2)

Now, let the number of errors detected in \((t, t+\Delta t)\) be proportional to the number of undetected faults, i.e.,

\[
j(t + \Delta t) - j(t) = b \{a - j(t)\} \Delta t,
\] (4.3)

where \( b \) is a constant of proportionality. Expression (4.3) leads to the differential equation.

\[
j'(t) = a b - b j(t).
\] (4.4)

Solving this for \( j(t) \), we get

\[
j(t) = a (1 - e^{-bt}).
\] (4.5)

Let \( \{J(t), \ t \geq 0\} \) be a counting process (number of errors in \((0,t]\)) that has independent increments so that the numbers of errors detected during disjoint time-intervals are independent. The difference between \( j(t) \) and \( J(t) \) is that the former is a deterministic number
whereas the latter is a random variable. Goel (1980) gave a good definition of an NHPP as we will mention next. The \( J(t) \) is an NHPP with intensity function \( \zeta(t) \) if

1) \( J(0) = 0 \);  
2) \( \{J(t), t \geq 0\} \) has independent increments;  
3) \( P\{2 \text{ or more increment in } (t, t+h)\} = o(h) \);  
4) \( P\{\text{exactly 1 increment in } (t, t+h)\} = \zeta(t) h + o(h) \).

If we put

\[
H(t) = \int_0^t \zeta(s) \, ds , \tag{4.6}
\]

then it can be shown that [Goel and Okumoto (1979)]

\[
P\{J(t) = n\} = (H(t))^n (n!)^{-1} \exp\{-H(t)\} , \quad n \geq 0. \tag{4.7}
\]

In other words, \( J(t) \) has a Poisson distribution with expected value \( H(t) \), which for \( t > 0 \), is the mean value function of the NHPP, indicating the expected cumulative number of errors detected up to testing time \( t \).

The deterministic model derived in (4.5) has been found to be a good descriptor of the software failure process when applied to actual data sets. For this reason we choose the mean value function to be

\[
H(t) = j(t) = a \left(1 - e^{-bt}\right) . \tag{4.8}
\]

Then the distribution of \( J(t) \) becomes

\[
P\{J(t) = n\} = \left[a \left(1 - e^{-bt}\right)\right]^n (n!)^{-1} \exp\{a(1 - e^{-bt})\} , \quad n = 0, 1, 2, \ldots . \tag{4.9}
\]

Note that
which is, again, a Poisson distribution. Moreover, let \( N(i) \) be the number of the remaining errors in the system at time \( t \). Then

\[
N(t) = J(\infty) - J(i) .
\]  

From (4.9) and (4.10), we get

\[
n(t) = E\{N(t)\} = a - H(t) .
\]

Putting \( J(\infty) = N \), we have

\[
\text{var}\{N(t)\} = \text{var}(N) + \text{var}\{J(t)\} - 2 \text{cov}\{J(t), N\}
= a + a \left(1 - e^{-bt}\right) - 2 a \left(1 - e^{-bt}\right)
= a e^{-bt} = n(t) .
\]  

The software reliability is the conditional survival probability of \( X_i \) on the condition that \( S_{i-1} = t \) is given by

\[
R(x|t) = P\{X_i > x \mid S_{i-1} = t\}
= \exp\left[-\{H(t + x) - H(t)\}\right] ; \quad t > 0 , \ x > 0 .
\]  

The last equation represents the probability that a software failure does not occur in \((t, t+x)\).

The mean time between failures (\( \mu \text{TBF} \)) is
\[ \mu_{TBF}(t) = \frac{1}{\xi(t)} \quad , \]

where \( \xi(t) \) is the intensity function of an NHPP. Equation (4.16) is called the instantaneous \( \mu_{TBF} \) at testing time \( t \).

There are two classes of NHPP models: finite failures and infinite failures. For the first kind of models the mean-value function must equal zero at time zero and must equal the expected number of errors \( a \) at infinite time, i.e.,

\[ H(0) = 0 \quad \text{and} \quad H(\infty) = a \quad . \]

(4.17)

For infinite failures models the mean value function at infinite time is infinity.

In the next Section we shall introduce a number of important software reliability growth models (SRGMs) which appear in the literature.

### 4.3 Software reliability growth models (SRGMs)

Assuming that the expected number of errors detected per unit testing time is proportional to the current residual fault content, then many SRGMs can be formulated as:

\[ \frac{dH(t)}{dt} = b(t) \left( a - H(t) \right) \quad ; \quad b(t) > 0, \ t \geq 0 \quad . \]

Or

\[ \frac{dH(t)}{a - H(t)} = b(t) \ dt \quad , \]

(4.18)

where \( b(t) \) is the error-detection rate per error at testing time \( t \).
By integrating both sides of (4.18) and evaluating the constant of integration using the condition \( H(0) = 0 \), we obtain

\[
\ln\{a - H(t)\} = - \int_0^t b(x) \, dx + \ln a .
\]

It then follows that

\[
H(t) = a \left\{ 1 - \exp\left( - \int_0^t b(x) \, dx \right) \right\} ; \quad t \geq 0 . \tag{4.19}
\]

### 4.3.1 Exponential (G&O) SRGM

If \( b(t) = b \) (constant), then we have the exponential SRGM proposed by Goel and Okumoto (1979) which describes a software failure-occurrence process in the testing. The mean value function is given by (4.8) as

\[
H(t) = m(t) = a \left( 1 - e^{-bx} \right) ; \quad a > 0, \quad b > 0 , \tag{4.20}
\]

with homogeneous error-detection rate \( b \). The failure intensity function for this model is obtained from (4.6) and (4.8) as

\[
\xi(t) = \frac{dH(t)}{dt} .
\]

\[
= abe^{-bt} . \tag{4.21}
\]

The expected number of errors remaining in the software at testing time \( t \) is given by (4.13). The software reliability is derived from (4.15) and (4.8) as

\[
R(x|t) = \exp\left[ -a \left( e^{-bt} - e^{-b(t+x)} \right) \right]
\]

\[
= \exp\left[ -ae^{-bt} \left( 1 - e^{-bx} \right) \right] , \tag{4.22}
\]

and the mean time between software failures is [see (4.16) and (4.21)].
Because this model is a two-parameter model with parameters \( a \) and \( b \) being the expected number of errors and the error detection rate, respectively, it is a relatively simple model and, therefore, widely used.

### 4.3.2 Modified exponential SRGM

In contrast to the previous case of the exponential SRGM, the detectability of an error is considered herein to be non-homogeneous over the testing period, since the kinds of errors detected early in the testing are usually different from those detected later on. Then, assuming that there are two types of error, of which Type 1 (Type 2) errors are easy (difficult) to detect, Yamada et al. (1985) proposed a non-homogeneous error-detection rate model in which error-detection processes for Types 1 and 2 errors are respectively described by the exponential SRGM of the last Section and called it the modified exponential SRGM.

The mean value function of the modified exponential SRGM is given by

\[
H(t) = m_p(t) = a \sum_{i=1}^{2} p_i (1 - e^{-b_i t} ) ,
\]

where

\[
0 < b_2 < b_1 < 1 , \quad \sum_{i=1}^{2} p_i = 1 , \quad 0 < p_i < 1 , \quad i = 1 , 2 ,
\]

and \( b_i \) is the error-detection rate per Type \( i \) error and \( p_i a \) is the expected initial error content of Type \( i \) error (\( i = 1 , 2 \) ). From (4.20), the error-detection rate per error for this model is given by applying (4.18) as follows
As before the failure intensity function is

\[ \xi(t) = a \sum_{i=1}^{2} b_i p_i e^{-b_i t} . \]  

(4.26)

The expected number of errors remaining in the software at testing time \( t \) is

\[ n(t) = a \sum_{i=1}^{2} p_i e^{-b_i t} . \]  

(4.27)

The software reliability is

\[ R(x|t) = \exp \left[ -a \sum_{i=1}^{2} p_i \left( e^{-b_i t} - e^{-b_i (t+x)} \right) \right] \]

\[ = \exp \left[ -a \sum_{i=1}^{2} p_i e^{-b_i t} (1 - e^{-b_i x}) \right] , \]  

(4.28)

and the mean time between software failures is

\[ \mu_{TBF}(t) = \left( a \sum_{i=1}^{2} b_i p_i e^{-b_i t} \right)^{-1} . \]  

(4.29)
4.3.3 S-shaped SRGM

In a software error removal process it should be assumed that a testing process consists of a software error-isolation process. If \( b(t) \), the error-detection rate per error, is given in the form

\[
b(t) = \frac{b^2 t}{1 + bt} = b - \frac{b}{1 + bt} \tag{4.30}
\]

then we have the delayed S-shaped SRGM proposed by Yamada et al. (1983). For such an error-detection process:

\[
H(t) = M(t) = a \left\{ 1 - (1 + bt) e^{bt} \right\} ; \quad a > 0, b > 0 \tag{4.31}
\]

which shows an S-shaped growth curve. The parameter \( b \) in equation (4.30) represents the failure-occurrence rate (the error-isolation rate). The failure intensity function for the delayed S-shaped SRGM is given by

\[
\xi(t) = ab^2 te^{-bt}. \tag{4.32}
\]

The expected number of errors remaining in the software at testing time \( t \) is

\[
n(t) = a(1 + bt) e^{-bt}. \tag{4.33}
\]

The software reliability is

\[
R(x|t) = \exp \left[ -a \left\{ (1 + bt) e^{-bt} - (1 + b(t + x)) e^{-b(t+x)} \right\} \right] \\
= \exp \left[ -ae^{-bt} \left\{ (1 + bt) - (1 + b(t + x)) e^{-bx} \right\} \right] \\
= \exp \left[ -ae^{-bt} \left\{ (1 + bt)(1 - e^{-bx}) - bxe^{-bx} \right\} \right] \tag{4.34}
\]

and the mean time between software failures is
\[ \mu TBF(t) = \frac{e^{bt}}{ab^2t} \quad (4.35) \]

### 4.3.4 Inflection S-shaped SRGM

This model was proposed by Ohba (1984a). It describes a software failure occurrence process with a mutual dependency of detected errors. In the error-detection process, the failures become detectable. For such a process, if in equation (4.18) we set

\[ b(t) = b \left\{ r + \left( 1 - r \right) \frac{H(t)}{a} \right\} , \quad (4.36) \]

where \( r \) is the error-detection rate per unit testing-effort expenditures, then the mean value function can be shown to be given by (see proof (1) in Appendix A)

\[ H(t) = I(t) = \frac{a \left( 1 - e^{-bt} \right)}{1 + ce^{-bt}} ; \quad a > 0, b > 0, c = (1 - r)/r > 0 , \quad (4.37) \]

which shows an S-shaped growth curve. The parameters \( a, b \) and \( c \) represent the expected number of errors in the system, the failure-detection rate and the inflection function respectively. The two parameters \( a \) and \( b \) will be estimated later in this chapter and hence it will be possible to estimate \( b(t) \) from the relation

\[ b(t) = \frac{b}{1 + ce^{-bt}} . \quad (4.38) \]

The failure intensity function is

\[ \xi(t) = \frac{abe^{-bt}(1+c)}{(1+ce^{-bt})^2} . \quad (4.39) \]

The expected number of errors remaining in the software at testing time \( t \) is
The software reliability is

\[
R(x|t) = \exp\left(-\frac{a(1-e^{-bt})}{1+ce^{-bt}}\right)
\]

and the mean time between software failures is

\[
\mu_{\text{TBF}}(t) = \frac{(1+ce^{-bt})^2e^{bt}}{ab(1+c)}.
\]

### 4.3.5 Testing-effort dependent SRGM

The testing-effort is measured by the amount of man-power, the CPU time, the number of executed test cases, and so on. Assume that the error detection rate per error is proportional to the current error content and the "constant" of proportionality is the current testing-effort expenditure. If in equation (4.18), we let

\[
b(t) = r \cdot w(t)
\]
where $r$ is the error-detection rate per unit testing-effort expenditure, the function $w(t)$ represents the testing-effort expenditure and is given [Yamada et al. (1986a, 1986b)] by the Weibull curve

$$w(t) = \alpha \beta \, t^{m-1} \, e^{-\beta \, t^m} \quad ; \quad \alpha > 0, \beta > 0, m > 0 \quad ,$$

(4.44)

where $\alpha$, $\beta$ and $m$ are constant parameters. The choices made in (4.43) and (4.44) result in the following testing-effort dependent SRGM

$$H(t) = T(t) = a \left[ 1 - \exp\{-rW(t)\} \right] \quad ; \quad a > 0, r > 0 \quad ,$$

(4.45)

where

$$W(t) = \int_0^t w(x) \, dx \quad .$$

(4.46)

In this case $a$ represents the total testing-effort required by testing. When $m = 1$ and $m = 2$, we have the exponential and Rayleigh testing effort functions, respectively. From equations (4.44) and (4.46), the total testing-effort in the testing time-interval $(0,t]$ is given by

$$W(t) = \alpha \left( 1 - e^{-t^m} \right) \quad .$$

(4.47)

From equations (4.45) and (4.47) we get

$$T(\infty) = a \left( 1 - e^{-\alpha} \right) \neq a \quad ,$$

(4.48)

which gives the expected number of errors to be eventually detected. The failure intensity function is

$$\xi(t) = arw(t)\exp\{-rW(t)\} \quad .$$

(4.49)

The expected number of errors remaining in the software at testing time $t$ is

$$n(t) = a \exp\{-rW(t)\} \quad .$$

(4.50)
The software reliability is

\[ R(x|t) = \exp[-a\{\exp(-rW(t)) - \exp(-rW(t+x))\}] \quad , \tag{4.51} \]

and the mean time between software failures is

\[ \mu_{TBF}(t) = \frac{\exp[rW(t)]}{arw(t)} \quad . \tag{4.52} \]

### 4.3.6 K-stage Erlangian (gamma) growth curve model

This model, called the K-model, was applied by Khoshgoftaar in (1988). He observed that the Goel and Okumoto (G&O) model and the S-shaped model could be described as special cases of a gamma function. The mean value function of this model is

\[ H(t) = \gamma(t) = a \left\{ 1 - e^{-bt} \sum_{j=0}^{K-1} \frac{(bt)^j}{j!} \right\} \quad , \tag{4.53} \]

where \( K \) is any positive integer. The error-detection rate per error for this model is

\[ b(t) = b \frac{(bt)^{K-1}}{(K-1)!} \sqrt{\sum_{j=0}^{K-1} \frac{(bt)^j}{j!}} \quad , \tag{4.54} \]

and the failure intensity function is

\[ \xi(t) = \frac{d\gamma(t)}{dt} = a b e^{-bt} \frac{(bt)^{K-1}}{(K-1)!} \quad . \tag{4.55} \]

The expected number of errors remaining in the software at testing time \( t \) is

\[ n(t) = ae^{-bt} \sum_{j=0}^{K-1} \frac{(bt)^j}{j!} \quad . \tag{4.56} \]
The software reliability is

\[ R(x|t) = \exp \left[ -a \left( e^{-bt} \sum_{j=0}^{K-1} \frac{(bt)^j}{j!} - e^{-b(t+x)} \sum_{j=0}^{K-1} \frac{(b(t+x))^j}{j!} \right) \right] \]

\[ = \exp \left[ -ae^{-bt} \left( \sum_{j=0}^{K-1} \frac{(bt)^j}{j!} - e^{bx} \sum_{j=0}^{K-1} \frac{(bt)^j}{j!} \right) \right] \quad , \tag{4.57} \]

and the mean time between software failures is

\[ \mu_{TBF}(t) = \frac{e^{bt} (K-1)!}{ab(bt)^{K-1}} \quad . \tag{4.58} \]

In the case of \( K = 1 \) and \( K = 2 \) we obtain the exponential SRGM, and the S-shaped SRGM, respectively. The K-model at \( K = 3 \) and \( K = 4 \) is a two-parameter model with parameters \( a \) and \( b \) being the expected number of errors and the error detection rate, respectively. The mean value function of the K-model at \( K = 3 \) is given by

\[ H(t) = a \left\{ 1 - \left(1 + \frac{1}{2} y \right) e^{-bt} \right\} \quad , \tag{4.59} \]

where

\[ y = bt(2 + bt) \quad . \]

And the mean value function of the K-model at \( K = 4 \) is given by

\[ H(t) = a \left\{ 1 - \left(1 + \frac{1}{6} y_1 \right) e^{-bt} \right\} \quad , \tag{4.60} \]

where

\[ y_1 = bt \left\{ 6 + 3bt + (bt)^2 \right\} \quad . \]
All the characteristics of the last two models are shown in Table 4.1. The two parameters \((a\) and \(b)\) of each case of the K-model will be estimated in Section 4.4.

### 4.3.7 Hyperexponential SRGM

According to Khoshgoftaar and Woodcock (1992), the general idea behind the hyperexponential model is that as the data is collected, it is divided into clusters based on some external information about the data (such as which module of the program caused the error). The mean value function of this model is

\[
H(t) = \sum_{i=1}^{n} a_i \left(1 - e^{-b_i t}\right) ,
\]

where \(n\) is the number of clusters of modules that have similar characteristics.

In this case the error-detection rate per error is

\[
b(t) = \frac{\sum_{i=1}^{n} a_i b_i e^{-b_i t}}{\sum_{i=1}^{n} a_i e^{-b_i t}} ,
\]

and the failure intensity function is

\[
\xi(t) = \sum_{i=1}^{n} a_i b_i e^{-b_i t} .
\]

The expected number of errors remaining in the software at testing time \(t\) is

\[
n(t) = \sum_{i=1}^{n} a_i e^{-b_i t} .
\]

The software reliability is
\[ R(x|t) = \exp \left[ -\sum_{i=1}^{n} a_i \left( e^{-b_i t} - e^{-b_i (t+x)} \right) \right] \]
\[ = \exp \left\{ -\sum_{i=1}^{n} a_i e^{-b_i t} \left( 1 - e^{-b_i x} \right) \right\}, \quad (4.65) \]

and the mean time between software failures is

\[ \mu_{TBF}(t) = \left( \sum_{i=1}^{n} a_i b_i e^{-b_i t} \right)^{-1}. \quad (4.66) \]

Notice that when there is only one cluster, \( n = 1 \), this model reduces to the Goel and Okumoto model.

The hyperexponential model has two parameters for each cluster \( i \), \( a_i \) is the expected number of errors in cluster \( i \), and \( b_i \) is the error-detection rate for the same cluster.

### 4.3.8 Littlewood SRGM

Littlewood (1981) proposed an NHPP model, which has the following mean-value function:

\[ H(t) = a \left\{ 1 - b^c (b+t)^{-c} \right\}. \quad (4.67) \]

The parameter \( a \) still stands for the expected number of errors in the software system, but the shape of the curve is determined by two parameters, \( b \) and \( c \).

The error-detection rate per error is

\[ b(t) = \frac{c}{b+t}. \quad (4.68) \]

and the failure intensity function is
The expected number of errors remaining in the software at testing time $t$ is

$$n(t) = ab^c(b + t)^{e-1}, \quad (4.70)$$

the software reliability is

$$R(x|t) = \exp[-ab^c\{1 - (b + (t + x))^{e-1}\}], \quad (4.71)$$

and the mean time between software failures is

$$\mu_{TBF}(t) = \frac{(b + t)^{e-1}}{acb^c}. \quad (4.72)$$

Notice that this model is not based on the exponential distribution but rather on the Pareto distribution.

### 4.3.9 Duane Model

The Duane model was derived from the hardware reliability area. Duane (1964) observed several hardware applications in which the rate of failure occurrence was in power law form in operating time. Crow (1974) assumed that this process was an NHPP process. The mean value function and the error-detection rate per error are respectively

$$H(t) = D(t) = a t^b, \quad (4.73)$$

and

$$b(t) = \frac{bt^{b-1}}{(1 - t^b)} \quad (4.74)$$

Also we have
\[ \xi(t) = a \cdot b \cdot t^{b-1} \quad , \quad (4.75) \]

\[ n(t) = a(1 - t^b) \quad , \quad (4.76) \]

\[ R(x|t) = \exp \left[ -a\left( (t+x)^b - t^b \right) \right] \quad , \quad (4.77) \]

and mean time between software failures

\[ \mu_{TBF}(t) = \frac{t^{1-b}}{ab} \quad . \quad (4.78) \]

### 4.3.10 Testing domain SRGM

Ohtera et al. (1990) proposed this model by defining a testing domain function

\[ u(t) = a \left( 1 - pe^{-\nu t} \right) \quad ; \quad a > 0 \ , \ \nu > 0 \ , \ 0 < p \leq 1 \quad , \quad (4.79) \]

where \( \nu \) is the testing-domain growth rate, \( p \) is the parameter representing the error distribution patterns in tested software. In equation (4.18), if we put \( b(t) = b \) and \( u(t) \) is substituted for \( a \), then we have (see proof (2) in Appendix A)

\[ H(t) = a \left\{ 1 - \frac{f - bs}{(v - b)} \right\} \quad , \quad (4.80) \]

where

\[ f = (\nu - b + bp)e^{-bt} \quad , \quad s = pe^{-\nu t} \quad \text{and} \quad \nu \neq b \quad . \]

In the case of \( \nu = b \) the mean value function will be in the form,

\[ H(t) = a \left[ 1 - (bpt + 1)e^{-bt} \right] \quad , \quad (4.81) \]
and the model characteristics, \( v \neq b \), are

\[
b(t) = \frac{b(f - vs)}{f - bs}, \quad (4.82)
\]

\[
\xi(t) = \frac{ab(f - vs)}{v - b}, \quad (4.83)
\]

\[
\eta(t) = \frac{a(f - bs)}{v - b}, \quad (4.84)
\]

\[
R(x|t) = \exp\left[-\frac{a}{v - b} \left\{ (v - b + bp)e^{-b(x)} - bpe^{-v(x)} - (v - b + bp)e^{-b(x)} - bpe^{-v(x)} \right\}\right]
\]

\[
= \exp\left[-\frac{a}{v - b} \left\{ f(1 - e^{-hx}) - s(1 - e^{-vx}) \right\}\right], \quad (4.85)
\]

and mean time between software failures

\[
\mu_{TBF}(t) = \frac{v - b}{ab(f - vs)}. \quad (4.86)
\]

### 4.3.11 Logistic and Gompertz SRGM

Deterministic SRGMs formulated by logistic and Gompertz growth curves have been widely used to estimate the error content [Yamada (1991)]. In Japan, some computer manufacturers and software houses have actually applied these models. The expected cumulative number of errors detected up to testing time \( t \) is given for the logistic growth curve model by

\[
H(t) = L(t) = \frac{\eta}{1 + me^{-\eta t}}; \quad \eta > 0, m > 0, p > 0, \quad (4.87)
\]
where \( k, p \) and \( m \) are constant parameters to be estimated. The parameter \( k \) is the expected initial error content of a software product, and the model characteristics are

\[
b(t) = \frac{p}{1 + me^{-pt}} , \quad (4.88)
\]

\[
\xi(t) = \frac{\eta p e^{-pt}}{(1 + me^{-pt})^2} , \quad (4.89)
\]

\[
n(t) = \frac{\eta me^{-pt}}{1 + me^{-pt}} , \quad (4.90)
\]

\[
R(x|t) = \exp\left(-\left(\frac{\eta}{1 + me^{-pt(x-t)}} - \frac{\eta}{1 + me^{-pt}}\right)\right)
\]

\[
= \exp\left[-\eta m\left(\frac{1 - e^{-px}}{(1 + me^{-pt})(e^{pt} + me^{-pt})}\right)\right] , \quad (4.91)
\]

and mean time between software failures

\[
\mu_{TBF}(t) = \frac{(1 + me^{-pt})^2 e^{pt}}{\eta pm} . \quad (4.92)
\]

For the Gompertz growth curve model, the expected cumulative number of errors detected up to testing time \( t \) is given by

\[
H(t) = G(t) = \eta a^{b^t} ; \quad \eta > 0, \ 0 < a < 1, \ 0 < b < 1 \quad , \quad (4.93)
\]

where \( k, c \) and \( b \) are constant parameters to be estimated. The parameter \( k \) is the expected initial error content of a software product. As before we have

\[
b(t) = \frac{b^t \ln a \ln b}{a^{-b^t} - 1} , \quad (4.94)
\]
\[ \xi(t) = \eta a^b b' \ln a \ln b, \quad (4.95) \]

\[ n(t) = \eta(b - a^b), \quad (4.96) \]

\[ R(x|t) = \exp\left\{ -\left( \eta a^{b(x-x)} - \eta a^b \right) \right\} \]
\[ = \exp\left\{ -\eta a^b \left( a^{b(x-x)} - 1 \right) \right\}, \quad (4.97) \]

and mean time between software failures

\[ \mu_{TBF}(t) = \frac{a^b b^{-t}}{\eta \ln a \ln b}. \quad (4.98) \]

For ease of reference, we summarize the key results of Section 4.3 in Table 4.1.
| SRGM Model                        | b(t) | ξ(t)           | n(t)           | R(x|t)                       | µTBF(t)   |
|----------------------------------|------|----------------|----------------|-----------------------------|-----------|
| Exponential SRGM                 | b    | abc<sup>-bt</sup> | ac<sup>-bt</sup> | exp<sup>{-ac<sup>-bt</sup>[1-e<sup>-bx</sup>]}<sup> </sup> | e<sup>bt</sup> / ab |
| If(t) = a<sup>{1-e<sup>-bt</sup>}</sup> |      |                |                |                             |           |
| Modified Exponential SRGM        |      |                |                |                             |           |
| mp(t) = a<sup>2</sup> Σ<sub>i=1</sub> pi(1-e<sup>-bi</sup>) |      |                |                |                             |           |
| S-shaped SRGM                    |      |                |                |                             |           |
| M(t) = a<sup>{1-(1+bt)e<sup>-bt</sup>}</sup> |      |                |                |                             |           |
| Inflection S-shaped SRGM         |      |                |                |                             |           |
| I(t) = a<sup>{1-e<sup>-bt</sup>}</sup> / (1+ce<sup>-bt</sup>) |      |                |                |                             |           |
| Testing-effort dependent SRGM    |      |                |                |                             |           |
| T(t) = a[1 - exp(-rW(t))]        |      |                |                |                             |           |

Table 4.1: Summary of the characteristics of different SRGMs.
<table>
<thead>
<tr>
<th>Table 4.1: Continued</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)πθερ</td>
</tr>
<tr>
<td>( \left( \frac{\pi\theta\epsilon}{\pi\theta\epsilon} \right)^{\frac{1}{(i)}} )</td>
</tr>
<tr>
<td>( \left( \frac{\pi\theta\epsilon}{\pi\theta\epsilon} \right)^{\frac{1}{(i)}} )</td>
</tr>
<tr>
<td>( \left( \frac{\pi\theta\epsilon}{\pi\theta\epsilon} \right)^{\frac{1}{(i)}} )</td>
</tr>
<tr>
<td>( \left( \frac{\pi\theta\epsilon}{\pi\theta\epsilon} \right)^{\frac{1}{(i)}} )</td>
</tr>
<tr>
<td>( \left( \frac{\pi\theta\epsilon}{\pi\theta\epsilon} \right)^{\frac{1}{(i)}} )</td>
</tr>
<tr>
<td>( \left( \frac{\pi\theta\epsilon}{\pi\theta\epsilon} \right)^{\frac{1}{(i)}} )</td>
</tr>
<tr>
<td>( \left( \frac{\pi\theta\epsilon}{\pi\theta\epsilon} \right)^{\frac{1}{(i)}} )</td>
</tr>
<tr>
<td>( \left( \frac{\pi\theta\epsilon}{\pi\theta\epsilon} \right)^{\frac{1}{(i)}} )</td>
</tr>
<tr>
<td>( \left( \frac{\pi\theta\epsilon}{\pi\theta\epsilon} \right)^{\frac{1}{(i)}} )</td>
</tr>
<tr>
<td>( \left( \frac{\pi\theta\epsilon}{\pi\theta\epsilon} \right)^{\frac{1}{(i)}} )</td>
</tr>
<tr>
<td><strong>SRGM Model</strong></td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Littlewood SRGM</td>
</tr>
<tr>
<td>Duane SRGM</td>
</tr>
<tr>
<td>Testing domain SRGM</td>
</tr>
<tr>
<td>Logistic SRGM</td>
</tr>
<tr>
<td>Gompertz SRGM</td>
</tr>
</tbody>
</table>

Table 4.1: Continued.
4.4 Estimation of the parameters of the SRGMs

It is necessary to estimate unknown parameters in the applied SRGMs using the observed test data. A set of test data is usually collected and recorded in one of two typical ways. The first one is called error-detection count data, see Figure 1.3. These are used when the project managers want to estimate the number of errors detected during a constant time-interval of testing, that is realization of random variables \( J(t_i) \), \( i = 1, 2, 3, \ldots, n \). For the error-detection count data, we want to predict the behavior of \( J(t_i) \) at a future time \( t_i \), \( i = n+1, n+2, \ldots \), using the observed data \( (t_i, y_i), i = 1, 2, \ldots, n \).

The second type of collected data is called failure-occurrence time data. These data, recorded as the times of successive software failures, that is, the realization of random variables \( S_i \), \( i = 1, 2, \ldots, n \), are the most detailed and desirable in terms of estimation, and are most often used in SRGMs based on the CPU time (or execution time) as a unit of testing time. Figure 1.4 shows this sort of data. The observed data \( s_i, i = 1, 2, \ldots, n \), can be converted into \( x_i \) by calculating \( x_i = s_i - s_{i-1} \). For this type of data, we want to predict the behavior at a future time \( s_i \) or the time-interval between failures \( X_i \), \( i = n+1, n+2, \ldots \), by using the observed data \( s_i \) or \( x_i, i = 1, 2, \ldots, n \).

In this section, we will derive the estimators for the parameters of each of the SRGMs that we are studying. This is important foundational work to enable us to perform the empirical evaluations of the models. In addition, we believe this is a valuable contribution in its own right as to our knowledge details of all these estimators are not available in the literature.

4.4.1 Estimation of the SRGMs parameters in case of Count Data

Suppose that the error-detection count data \( (t_i, y_i), i = 1, 2, \ldots, n \), are observed during the testing phase. Then the likelihood function for the unknown parameters in an NHPP model with \( H(t) \) is given by
\[ L(a, \theta | t, y) = P \{ J(t_1) = y_1, J(t_2) = y_2, \ldots, J(t_n) = y_n \} \]
\[ = \prod_{i=1}^{n} \frac{[H(t_i) - H(t_{i-1})]}{(y_i - y_{i-1})!} \exp \left[ -\frac{[H(t_i) - H(t_{i-1})]}{} \right] \tag{4.99} \]

where
\[ t_0 = 0, \ y_0 = 0 \]
\[ \theta = (\theta_1, \theta_2, \ldots, \theta_n) \] is the \( n \) model parameters,
\[ t = (t_1, t_2, \ldots, t_n), \]
\[ S = (s_1, s_2, \ldots, s_n), \]
\[ y = (y_1, y_2, \ldots, y_n). \]

Taking the natural logarithm of both sides of equation (4.99) and equating its partial derivatives with respect to the unknown parameters to zero, we get
\[ \frac{\partial \ln L(a, \theta | t, y)}{\partial a} = 0 \tag{4.100} \]
and
\[ \frac{\partial \ln L(a, \theta | t, y)}{\partial \theta_i} = 0, \quad i = 1, 2, \ldots, n. \tag{4.101} \]

Then, the \((n+1)\) maximum likelihood estimates \( \hat{a} \) and \( \hat{\theta}_i \), \( i = 1, 2, \ldots, n \) can be obtained by solving equations (4.100) and (4.101) simultaneously.

Now we apply this maximum likelihood technique to the SRGMs mentioned above.

### 4.4.1.1 K-Stage Erlangian (gamma) model

As we mentioned in Section 4.3, equation (4.53), the mean value function of this model is
And the likelihood function is given by

\[
L_i(a, b \mid t, y) = \prod_{i=1}^{n} \frac{\{\gamma (t_i) - \gamma (t_{i-1})\}^{y_i - y_{i-1}}}{(y_i - y_{i-1})!} \exp \left[ -\{\gamma (t_i) - \gamma (t_{i-1})\} \right] 
\]

\[
= \prod_{i=1}^{n} \left[ a \{1 - e^{-bt_i} \sum_{j=0}^{K-1} \frac{(bt_i)^j}{j!} \} - a \{1 - e^{-bt_{i-1}} \sum_{j=0}^{K-1} \frac{(bt_{i-1})^j}{j!} \} \right]^{y_i - y_{i-1}} 
\]

\[
- \{y_i - y_{i-1}\}! \exp \left[ -a \{1 - e^{-bt_i} \sum_{j=0}^{K-1} \frac{(bt_i)^j}{j!} \} + a \{1 - e^{-bt_{i-1}} \sum_{j=0}^{K-1} \frac{(bt_{i-1})^j}{j!} \} \right].
\]

(4.103)

Taking the natural logarithm of both sides, we get

\[
\ell_i(a, b \mid t, y) = \ell_i = \ln L_i(a, b \mid t, y)
\]

\[
= \sum_{i=1}^{n} \left[ (y_i - y_{i-1}) \ln a + (y_i - y_{i-1}) \ln \left( e^{bt_i} \sum_{j=0}^{K-1} \frac{(bt_i)^j}{j!} - e^{bt_{i-1}} \sum_{j=0}^{K-1} \frac{(bt_{i-1})^j}{j!} \right) \right]
\]

\[
- \sum_{i=1}^{n} \ln(y_i - y_{i-1}) - \sum_{i=1}^{n} a \left( e^{bt_i} \sum_{j=0}^{K-1} \frac{(bt_i)^j}{j!} - e^{bt_{i-1}} \sum_{j=0}^{K-1} \frac{(bt_{i-1})^j}{j!} \right).
\]

(4.104)

Differentiating with respect to \(a\) and then equating to zero, we get

\[
\frac{\partial \ell_i}{\partial a} = \frac{\sum_{i=1}^{n} (y_i - y_{i-1})}{a} - \sum_{i=1}^{n} a \left( e^{bt_i} \sum_{j=0}^{K-1} \frac{(bt_i)^j}{j!} - e^{bt_{i-1}} \sum_{j=0}^{K-1} \frac{(bt_{i-1})^j}{j!} \right) = 0,
\]

and so
\[
y_n \frac{a}{a} = \sum_{i=1}^{n} \left\{ e^{bt_i} \sum_{j=0}^{K-1} \frac{(bt_{i-1})^j}{j!} - e^{-bt_i} \sum_{j=0}^{K-1} \frac{(bt_i)^j}{j!} \right\}.
\]  \hspace{1cm} (4.105)

Since
\[
\sum_{i=1}^{n} \left\{ e^{-bt_i} \sum_{j=0}^{K-1} \frac{(bt_i)^j}{j!} - e^{-bt_i} \sum_{j=0}^{K-1} \frac{(bt_i)^j}{j!} \right\}
= e^{-bt_0} \sum_{j=0}^{K-1} \frac{(bt_0)^j}{j!} - e^{-bt_1} \sum_{j=0}^{K-1} \frac{(bt_1)^j}{j!} + e^{-bt_1} \sum_{j=0}^{K-1} \frac{(bt_1)^j}{j!} - e^{-bt_2} \sum_{j=0}^{K-1} \frac{(bt_2)^j}{j!} + \ldots \n+ e^{-bt_{n-1}} \sum_{j=0}^{K-1} \frac{(bt_{n-1})^j}{j!} - e^{-bt_n} \sum_{j=0}^{K-1} \frac{(bt_n)^j}{j!}
= 1 - e^{-bt_n} \sum_{j=0}^{K-1} \frac{(bt_n)^j}{j!},
\]  \hspace{1cm} (4.106)

so that finally we get
\[
a = \frac{y_n}{1 - e^{-bt_n} \sum_{j=0}^{K-1} \frac{(bt_n)^j}{j!}}.
\]  \hspace{1cm} (4.107)

Now, differentiate (4.104) with respect to \(b\). Then equate to zero and denote the expression
\[
e^{-bt_i} \sum_{j=0}^{K-1} \frac{(bt_{i-1})^j}{j!} - e^{-bt_1} \sum_{j=0}^{K-1} \frac{(bt_1)^j}{j!}
\]
by the symbol "Don". Then, the following expressions are obtained
\[
\frac{\partial \ell_i}{\partial b} = \sum_{i=1}^{n} \frac{y_i - y_{i-1}}{\text{Don}} \frac{\partial}{\partial b} (\text{Don}) - \sum_{i=1}^{n} \left\{ \frac{a}{\partial b} (\text{Don}) \right\}
= \sum_{i=1}^{n} \left[ \left\{ \frac{y_i - y_{i-1}}{\text{Don}} - a \right\} \frac{\partial}{\partial b} (\text{Don}) \right] = 0 .
\]  \hspace{1cm} (4.108)
\[
\frac{\partial}{\partial b} (\text{Don}) = e^{-bt_{i-1}} \sum_{j=0}^{K-1} j b^{j-1} (t_{i-1})^j \frac{j!}{j!} - t_{i-1} e^{-bt_{i-1}} \sum_{j=0}^{K-1} (bt_{i-1})^j \frac{j!}{j!} \\
- e^{-bt_{i-1}} \sum_{j=0}^{K-1} j b^{j-1} (t_{i})^j \frac{j!}{j!} + t_{i} e^{-bt_{i}} \sum_{j=0}^{K-1} (bt_{i})^j \frac{j!}{j!} \\
= t_{i-1} e^{-bt_{i-1}} \frac{\sum_{j=1}^{K-1} (bt_{i-1})^j \frac{1}{j!} - \sum_{j=0}^{K-1} (bt_{i-1})^j \frac{j!}{j!}}{t_{i-1} e^{-bt_{i-1}} \frac{\sum_{j=1}^{K-1} (bt_{i})^j \frac{1}{j!} - \sum_{j=0}^{K-1} (bt_{i})^j \frac{j!}{j!}} \\
+ t_{i} e^{-bt_{i}} \frac{\sum_{j=0}^{K-1} (bt_{i})^j \frac{1}{j!} - \sum_{j=1}^{K-1} (bt_{i})^j \frac{j!}{j!}}{t_{i} e^{-bt_{i}} \frac{\sum_{j=0}^{K-1} (bt_{i})^j \frac{1}{j!} - \sum_{j=1}^{K-1} (bt_{i})^j \frac{j!}{j!}}} \\
= t_{i} e^{-bt_{i}} \frac{(bt_{i})^{K-1}}{(K-1)!} - t_{i-1} e^{-bt_{i-1}} \frac{(bt_{i-1})^{K-1}}{(K-1)!} , \\
(4.109)
\]

then

\[
\sum_{i=1}^{n} \frac{\partial}{\partial b} (\text{Don}) = \sum_{i=1}^{n} \left\{ t_{i} e^{-bt_{i}} \frac{(bt_{i})^{K-1}}{(K-1)!} - t_{i-1} e^{-bt_{i-1}} \frac{(bt_{i-1})^{K-1}}{(K-1)!} \right\} \\
= t_{n} \frac{(bt_{n})^{K-1}}{(K-1)!} e^{-bt_{n}} . \\
(4.110)
\]

From (4.108), we have

\[
\frac{\partial t_{i}}{\partial b} = \sum_{i=1}^{n} y_{i} - y_{i-1} \frac{y_{n} t_{n} (bt_{n})^{K-1} e^{-bt_{n}}}{(K-1)! \left\{ 1 - e^{-bt_{n}} \sum_{j=0}^{K-1} (bt_{n})^j \frac{j!}{j!} \right\}} \\
\]

Therefore
If we put \(K = 1\) and 2 in the likelihood equations (4.107) and (4.111), we obtain the likelihood equations for the exponential and the S-shaped SRGMs, respectively. Hence, we obtain estimates for the parameters \(a\) and \(b\) in each case. That is for \(K = 1\), the exponential model, the MLEs of \(a\) and \(b\) can be obtained by solving the following two equations iteratively,

\[
a = \frac{y_n}{1 - e^{-bt_n}}
\]

and

\[
y_n t_n e^{-bt_n} \frac{n}{1 - e^{-bt_n}} = \sum_{i=1}^{n} \frac{(y_i - y_{i-1})(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}})}{e^{bt_{i-1}} - e^{-bt_i}}.
\]

When \(K = 2\), the delayed S-shaped model, the MLEs equations that should be solved are

\[
a = \frac{y_n}{1 - (1 + bt_n)e^{-bt_n}}
\]

and

\[
y_n t_n^2 e^{-bt_n} \frac{n}{1 - (1 + bt_n)e^{-bt_n}} = \sum_{i=1}^{n} \frac{(y_i - y_{i-1})(t_i^2 e^{-bt_i} - t_{i-1}^2 e^{-bt_{i-1}})}{(1 + bt_{i-1})e^{-bt_{i-1}} - (1 + bt_i)e^{-bt_i}}.
\]

The situations of \(K = 3\) and 4 are shown in Table 4.2.

**4.4.1.2 Modified exponential model**

The likelihood function for this model, using equation (4.24), is given by
\[ L_2(a, \theta \mid t, y) = \prod_{i=1}^{n} \left\{ a \sum_{i=1}^{2} p_i \left( e^{-b_{t_j-1}^i} - e^{-b_{t_j}^i} \right) \right\}^{y_j-y_j-1} \frac{1}{(y_j-y_{j-1})!} \exp \left\{ -a \sum_{i=1}^{2} p_i \left( e^{-b_{t_j-1}^i} - e^{-b_{t_j}^i} \right) \right\} . \]

Taking the natural logarithm of both sides in the last equation, we have

\[ \ln L_2(a, \theta \mid t, y) = \ell_2 = \sum_{j=1}^{n} \left( y_j - y_{j-1} \right) \ln \left\{ a \sum_{i=1}^{2} p_i \left( e^{-b_{t_j-1}^i} - e^{-b_{t_j}^i} \right) \right\} \]
\[ - \sum_{j=1}^{n} \ln(y_j - y_{j-1}) - \sum_{j=1}^{n} \left\{ a \sum_{i=1}^{2} p_i \left( e^{-b_{t_j-1}^i} - e^{-b_{t_j}^i} \right) \right\} . \]

(4.116)

Differentiate both sides of (4.116) with respect to \( a \) and equate to zero. The likelihood equations will be

\[ 0 = \frac{\partial \ell_2}{\partial a} = \sum_{j=1}^{n} \left( y_j - y_{j-1} \right) \frac{1}{a} - \sum_{j=1}^{n} \sum_{i=1}^{2} p_i \left( e^{-b_{t_j-1}^i} - e^{-b_{t_j}^i} \right) . \]

(4.117)

Equation (4.117) can be easily reduced to

\[ \frac{Y_n}{a} - \sum_{i=1}^{2} p_i \left( 1 - e^{-b_{t_n}^i} \right) = 0 , \]

and then, the MLE of \( a \) is

\[ a = \frac{Y_n}{\sum_{i=1}^{2} p_i \left( 1 - e^{-b_{t_n}^i} \right)} . \]

(4.118)

For the MLE of \( b \), we have

\[ \frac{\partial \ell_2}{\partial b} = \sum_{j=1}^{n} \frac{\left( y_j - y_{j-1} \right) \sum_{i=1}^{2} p_i \left( t_{j} e^{-b_{t_j}^i} - t_{j-1} e^{-b_{t_{j-1}}^i} \right)}{\sum_{i=1}^{2} p_i \left( e^{-b_{t_j-1}^i} - e^{-b_{t_j}^i} \right)} - a \sum_{i=1}^{2} p_i t_n e^{-b_{t_n}^i} = 0 . \]

(4.119)
Substituting for \(a\) from (4.118) in (4.119), we get

\[
\frac{y_n t_n \sum_{i=1}^{2} p_i e^{-b_i t_i^n}}{\sum_{i=1}^{2} p_i (1 - e^{-b_i t_i^n})} = \sum_{j=1}^{a} \left( y_j - y_{j-1} \right) \sum_{i=1}^{2} p_i \left( t_j e^{-b_i t_j} - t_{j-1} e^{-b_i t_{j-1}} \right).
\]  
(4.120)

\subsection{4.4.1.3 Inflection S-shaped model}

Consider equation (4.37) in Section 4.3, the likelihood function is

\[
L_3(a, \theta | t, y) = \prod_{i=1}^{n} \left\{ \frac{1 - e^{-b_i t_i}}{1 + ce^{b_i t_i}} - \frac{1 - e^{-b_{i-1} t_i}}{1 + ce^{b_{i-1} t_i}} \right\}^{y_i - y_{i-1}} \left\{ (y_i - y_{i-1}) \right\}^{-1} \exp \left\{ -a \left( \frac{1 - e^{-b_i t_i}}{1 + ce^{-b_i t_i}} - \frac{1 - e^{-b_{i-1} t_i}}{1 + ce^{-b_{i-1} t_i}} \right) \right\}.
\]  
(4.121)

By taking the natural logarithm of both sides, we get

\[
\ell_3 = \ln \left\{ L_3(a, \theta | t, y) \right\} = \sum_{i=1}^{n} (y_i - y_{i-1}) \ln \left( \frac{1 - e^{-b_i t_i}}{1 + ce^{-b_i t_i}} - \frac{1 - e^{-b_{i-1} t_i}}{1 + ce^{-b_{i-1} t_i}} \right) \]

\[
- \sum_{i=1}^{n} \ln (y_i - y_{i-1}) - \sum_{i=1}^{n} a \left( \frac{1 - e^{-b_i t_i}}{1 + ce^{-b_i t_i}} - \frac{1 - e^{-b_{i-1} t_i}}{1 + ce^{-b_{i-1} t_i}} \right).
\]  
(4.122)

Differentiate with respect to \(a\) and equate to zero. Then the following equation is obtained

\[
\frac{\partial \ell_3}{\partial a} = \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) - \sum_{i=1}^{n} a \left( \frac{1 - e^{-b_i t_i}}{1 + ce^{-b_i t_i}} - \frac{1 - e^{-b_{i-1} t_i}}{1 + ce^{-b_{i-1} t_i}} \right) = 0.
\]  
(4.123)

The second term of (4.123) is equal to
Solving (4.123) in \(a\), we get
\[
a = \frac{y_n(1 + ce^{-bt_n})}{1 - e^{-bt_n}}.
\] (4.124)

Differentiate (4.122) with respect to \(b\) so as to obtain
\[
\frac{\partial l_3}{\partial b} = \frac{\partial}{\partial b} \sum_{i=1}^{a} (y_i - y_{i-1}) \ln \left( \frac{1 - e^{-bi}}{1 + ce^{-bi}} \right) - \frac{\partial}{\partial b} \sum_{i=1}^{a} \left( 1 + c \right) \frac{e^{-bi} - e^{-bi-1}}{(1 + ce^{-bi})(1 + ce^{-bi-1})}.
\] (4.125)

Perform the differentiation on the first term, to get
\[
\frac{\partial}{\partial b} \left( \frac{1 - e^{-bi}}{1 + ce^{-bi}} \right) = \frac{\partial}{\partial b} \left[ \frac{(1 + c)(e^{-bi-1} - e^{-bi})}{(1 + ce^{-bi})(1 + ce^{-bi-1})} \right].
\] (4.126)

After simple calculations, we can rewrite (4.126) as
\[
\frac{t_i e^{-bi} - t_{i-1} e^{-bi-1}}{t_i e^{-bi} - t_{i-1} e^{-bi-1} + ct_i e^{-bi} + ct_{i-1} e^{-bi-1}}.
\] (4.127)

For the second term in (4.125), after ordinary calculations, we have
\[
\sum_{i=1}^{a} \left[ y_n \frac{1 + ce^{-bi}}{1 + ce^{-bi}} \left( \frac{1 - e^{-bi}}{1 + ce^{-bi}} - \frac{1 - e^{-bi-1}}{1 + ce^{-bi-1}} \right) \right] = \frac{y_n t_n e^{-bt_n} (1 + c)}{(1 - e^{-bt_n})(1 + ce^{-bt_n})}.
\] (4.128)

Substitute (4.127) and (4.128) in (4.125) and equate it to zero, so as to finally obtain
\[
\frac{(1 + c)y_n t_n e^{-bl_n}}{(1 - e^{-bl_n})} \left( \frac{t_i e^{-bl_i} + ct_i e^{-bl_i} + ct_{i-1} e^{-bl_{i-1}}}{1 + ce^{-bl_{i-1}}} \right) \\
\sum_{i=1}^{n}(y_i - y_{i-1}) \left( \frac{t_i e^{-bl_i} - t_{i-1} e^{-bl_{i-1}}}{1 + ce^{-bl_{i-1}}} \right) + ct_{i-1} e^{-bl_{i-1}}
\] (4.129)

4.4.1.4 Testing-effort dependent model

In this case the likelihood function is given by

\[
L_{d}(a, \theta \mid t, y) = \prod_{i=1}^{n} \frac{\left\{ a \left( e^{-rW(t_{i-1})} - e^{-rW(t_i)} \right) \right\}^{y_i - y_{i-1}} \cdot \exp \left\{-a \left( e^{-rW(t_{i-1})} - e^{-rW(t_i)} \right) \right\}}{(y_i - y_{i-1})!}
\]

We proceed to get the likelihood equations. Then

\[
\ell_{d} = \ln L_{d}(a, \theta \mid t, y),
\]

\[
\frac{\partial \ell_{d}}{\partial a} = \left[ \sum_{i=1}^{n}(y_{i} - y_{i-1}) \right] / a - \sum_{i=1}^{n} \left( e^{-rW(t_{i-1})} - e^{-rW(t_i)} \right) = 0,
\] (4.130)

where the second term can be written as

\[
1 - e^{-rW(t_{n})}.
\]

Substitute in (4.130) to obtain

\[
a = \frac{y_n}{1 - e^{-rW(t_{n})}}.
\] (4.131)

\[
\frac{\partial \ell_{d}}{\partial r} = \sum_{i=1}^{n} (y_{i} - y_{i-1}) \{ [W(t_{i}) e^{-rW(t_i)} - W(t_{i-1}) e^{-rW(t_{i-1})}] \\
\cdot \left( e^{-rW(t_{i-1})} - e^{-rW(t_i)} \right) \}^{i} - a W(t_n) e^{-rW(t_n)} = 0
\] (4.132)

Substituting (4.131) in (4.132), we obtain
4.4.1.5 Hyperexponential model

For this model, the mean value function was given by equation (4.61). Then the likelihood function is

\[
L_5(a, b \mid t, y) = L_5
\]

\[
= \prod_{i=1}^{n} \left[ \sum_{j=1}^{m} a_j \left( 1 - e^{-b_j t_i} \right) - \sum_{j=1}^{m} a_j \left( 1 - e^{-b_j (t_{i-1})} \right) \right] ^ {y_i - y_{i-1}} \left( y_i - y_{i-1} \right)^{-1}
\]

\[
\exp \left[ - \left( \sum_{j=1}^{m} a_j \left( 1 - e^{-b_j t_i} \right) - \sum_{j=1}^{m} a_j \left( 1 - e^{-b_j (t_{i-1})} \right) \right) \right].
\]

Therefore

\[
\ell_5 = \ln L_5 =
\]

\[
\sum_{i=1}^{n} (y_i - y_{i-1}) \ln \left[ \sum_{j=1}^{m} a_j \left( 1 - e^{-b_j t_i} \right) - \sum_{j=1}^{m} a_j \left( 1 - e^{-b_j (t_{i-1})} \right) \right] - \sum_{i=1}^{n} \ln (y_i - y_{i-1})
\]

\[
- \sum_{i=1}^{n} \left( \sum_{j=1}^{m} a_j \left( 1 - e^{-b_j t_i} \right) - \sum_{j=1}^{m} a_j \left( 1 - e^{-b_j (t_{i-1})} \right) \right).
\]

Put

\[
u = \sum_{j=1}^{m} a_j \left( 1 - e^{-b_j t_i} \right) - \sum_{j=1}^{m} a_j \left( 1 - e^{-b_j (t_{i-1})} \right) = \sum_{j=1}^{m} a_j \left( e^{-b_j t_{i-1}} - e^{-b_j t_i} \right),
\]

and hence obtain

\[
\frac{\partial u}{\partial a_j} = \sum_{j=1}^{m} \left( e^{-b_j t_{i-1}} - e^{-b_j t_i} \right).
\]
Substitute from (4.136) in (4.135), differentiate with respect to \(a\) and \(b\) and equate to zero. The result is

\[
\frac{\partial \ell_5}{\partial a_j} = \sum_{i=1}^{n} \left\{ \frac{(y_i - \bar{y}) \sum_{j=1}^{m} \left( e^{-b_{j_i+t}} - e^{-b_{j_i}} \right)}{\sum_{j=1}^{m} a_j \left( e^{-b_{j_i+t}} - e^{-b_{j_i}} \right)} \right\} - \sum_{j=1}^{m} \left( e^{-b_{j_i+1}} - e^{-b_{j_i+t}} \right)
\]

and

\[
\frac{\partial \ell_5}{\partial b_j} = \sum_{i=1}^{n} \frac{(y_i - \bar{y}) \left[ \sum_{j=1}^{m} \left( e^{-b_{j_i+t}} - e^{-b_{j_i}} \right) \right]}{\sum_{j=1}^{m} a_j \left( e^{-b_{j_i+t}} - e^{-b_{j_i}} \right)} - \frac{\partial \sum_{i=1}^{n} \sum_{j=1}^{m} a_j \left( e^{-b_{j_i+1}} - e^{-b_{j_i+t}} \right)}{\partial b_j} = 0.
\] (4.137)

If \(m = 1\), we have

\[
\frac{\partial \ell_5}{\partial a_1} = \sum_{i=1}^{n} \left\{ \frac{(y_i - \bar{y}) \left( e^{-b_{j_i+t}} - e^{-b_{j_i}} \right)}{a_1 \left( e^{-b_{j_i+t}} - e^{-b_{j_i}} \right)} - \left( e^{-b_{j_i+1}} - e^{-b_{j_i+t}} \right) \right\}
\]

\[
= \frac{y_n - 1}{a_1} e^{-b_{j_i+t}} = 0,
\]

and so

\[
a_1 = \frac{y_n}{1 - e^{-b_{j_i+t}}},
\] (4.138)

Similarly

\[
\frac{\partial \ell_5}{\partial b_1} = \sum_{i=1}^{n} a_1 \left[ e^{-b_{j_i+t}} - e^{-b_{j_i}} \right] \left( \frac{(y_i - \bar{y})}{a_1 \left( e^{-b_{j_i+t}} - e^{-b_{j_i}} \right)} - 1 \right) = 0.
\]

and therefore
\[
\frac{y_n t_n e^{-b t_{1n}}}{1 - e^{-b t_{1n}}} = \sum_{i=1}^{n} \left( \frac{y_i - y_{i-1}}{t_i e^{-b t_i} - t_{i-1} e^{-b t_{i-1}}} \right),
\]

which is the same result as in the case of the exponential model due to Goel and Okumoto.

If \( m = 2 \), we will have the four parameters case. By considering the likelihood function (4.135), the likelihood equations are

\[
\frac{\partial L}{\partial a_1} = \sum_{i=1}^{n} \left( e^{-b t_{1i-1}} - e^{-b t_{1i}} \right) \left\{ \frac{2 (y_i - y_{i-1})}{\sum_{j=1}^{\lambda} (e^{-b t_{1j-1}} - e^{-b t_j})} - 1 \right\} = 0.
\]

\[
\frac{\partial L}{\partial a_2} = \sum_{i=1}^{n} \left( e^{-b t_{1i-1}} - e^{-b t_{1i}} \right) \left\{ \frac{2 (y_i - y_{i-1})}{\sum_{j=1}^{\lambda} (e^{-b t_{1j-1}} - e^{-b t_j})} - 1 \right\} = 0.
\]

\[
\frac{\partial L}{\partial b_1} = \sum_{i=1}^{n} \left( t_i e^{-b t_{1i}} - t_{i-1} e^{-b t_{1i-1}} \right) \left\{ \frac{2 (y_i - y_{i-1})}{\sum_{j=1}^{\lambda} (e^{-b t_{1j-1}} - e^{-b t_j})} - 1 \right\} = 0.
\]

\[
\frac{\partial L}{\partial b_2} = \sum_{i=1}^{n} \left( t_i e^{-b t_{2i}} - t_{i-1} e^{-b t_{2i-1}} \right) \left\{ \frac{2 (y_i - y_{i-1})}{\sum_{j=1}^{\lambda} (e^{-b t_{2j-1}} - e^{-b t_j})} - 1 \right\} = 0.
\]

The estimates \( \hat{a}_1, \hat{a}_2, \hat{b}_1 \), and \( \hat{b}_2 \) could be evaluated by solving the last four equations iteratively.

### 4.4.1.6 Littlewood NHPP model

The mean value function of the three parameters Littlewood SRGM, was given by (4.67). The likelihood function in this case is
\[ L_\theta(a, t | y) = \prod_{i=1}^{n} \left[ \exp \left( -ab^c \left( b + t_{i-1} \right)^c - \left( b + t_i \right)^c \right) \right]^{y_i - y_{i-1}} \cdot \exp \left[ -ab^c \left( b + t_{i-1} \right)^c - \left( b + t_i \right)^c \right] \] (4.144)

Take the natural logarithm of both sides of (4.144), to get

\[ \ell_\theta = \ln L_\theta(a, t | y) = \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) \left[ \ln \left( b + t_{i-1} \right)^c - \left( b + t_i \right)^c \right] \]

\[-\sum_{i=1}^{n} \ln \left( y_i - y_{i-1} \right) \exp \left[ -ab^c \left( b + t_{i-1} \right)^c - \left( b + t_i \right)^c \right] \] (4.145)

Differentiate (4.145) with respect to \( a \) and equate to zero. The result is

\[ \frac{\partial \ell_\theta}{\partial a} = \sum_{i=1}^{n} \left[ \frac{y_i - y_{i-1}}{a} - b^c \left( b + t_{i-1} \right)^c - \left( b + t_i \right)^c \right] = 0. \]

Therefore

\[ a = \frac{y_n}{1 - b^c \left( b + t_n \right)^c} = \frac{y_n \left( b + t_n \right)^c}{\left( b + t_n \right)^c - b^c}. \] (4.146)

Similarly differentiate (4.145) with respect to \( b \) and equate to zero to produce the following

\[ \frac{\partial \ell_\theta}{\partial b} = \sum_{i=1}^{n} \frac{\left( y_i - y_{i-1} \right) c}{b} - c \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) \frac{\left( b + t_i \right)^{c+1} - \left( b + t_{i-1} \right)^{c+1}}{\left( b + t_i \right)^{c+1} \left( b + t_{i-1} \right) - \left( b + t_{i-1} \right)^{c+1} \left( b + t_i \right)} \]

\[-c \frac{y_n \left( b + t_n \right)^c}{\left( b + t_n \right)^c - b^c} \frac{c}{\left( b + t_n \right)^{c+1}} \]

\[ = \frac{c}{b} \frac{y_n}{1 - b^c \left( b + t_n \right)^c} - c \sum_{i=1}^{n} \frac{\left( y_i - y_{i-1} \right) \left( b + t_i \right)^{c+1} - \left( b + t_{i-1} \right)^{c+1}}{\left( b + t_i \right)^{c+1} \left( b + t_{i-1} \right) - \left( b + t_{i-1} \right)^{c+1} \left( b + t_i \right)} \]

\[-\frac{y_n c}{\left( b + t_n \right)^{c+1} - b^c \left( b + t_n \right)}. \] (4.147)

Equating (4.147) to zero, we get
Finally, differentiate with respect to \( c \), to obtain

\[
\frac{\partial \ell_6}{\partial c} = 
\sum_{i=1}^{n} \left( y_i - y_{i-1} \right) \ln b + \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) \frac{(b + t_{i-1})^c \ln(b + t_i) - (b + t_i)^c \ln(b + t_{i-1})}{(b + t_i)^c - (b + t_{i-1})^c} 
- \frac{\partial}{\partial c} \left[ a(1 - (b + t_n)^c) \right] \tag{4.148}
\]

Differentiating the last term of (4.148), we have

\[
y_n \left( \frac{\ln(b + t_n)}{(b + t_n)^c - b^c} - \ln b \right) = 
\sum_{i=1}^{n} \left( y_i - y_{i-1} \right) \frac{(b + t_{i-1})^c \ln(b + t_i) - (b + t_i)^c \ln(b + t_{i-1})}{(b + t_i)^c - (b + t_{i-1})^c}, \tag{4.149}
\]

### 4.4.1.7 Duane model

The Duane model is an infinite model with mean value function given by (4.73), for which the likelihood function is

\[
L_7(a, \theta | t, y) = \prod_{i=1}^{n} \frac{a(t_i^b - t_{i-1}^b)}{(y_i - y_{i-1})} \exp \left[ a(t_i^b - t_{i-1}^b) \right]. \tag{4.150}
\]

Proceed as before. Then obtain

\[
\frac{\partial \ell_7}{\partial a} = \sum_{i=1}^{n} \frac{(y_i - y_{i-1})}{a} - \sum_{i=1}^{n} (t_i^b - t_{i-1}^b) = 0.
\]
\[ \frac{y_n}{a} - t_n^b = 0 , \]
and therefore
\[ a = \frac{y_n}{t_n^b} . \]  

(4.151)

Similarly, we obtain the equation
\[ \frac{\partial \ell}{\partial b} = \sum_{i=1}^{n} \frac{y_i - y_{i-1}}{t_i^b - t_{i-1}^b} (t_i^b \ln t_i - t_{i-1}^b \ln t_{i-1}) - y_n \ln t_n = 0 , \]

which can be rearranged as
\[ y_n \ln t_n = \sum_{i=1}^{n} \frac{y_i - y_{i-1}}{t_i^b - t_{i-1}^b} (t_i^b \ln t_i - t_{i-1}^b \ln t_{i-1}) . \]  

(4.152)

### 4.4.1.8 The Gompertz growth curve model

The form of the mean value function is given by equation (4.93) and the parameters \( \eta, a \) and \( b \) would be estimated. The logarithm of the likelihood function \( \ell_8 \) in this case is
\[
\ell_8 = \sum_{i=1}^{n} (y_i - y_{i-1}) \ln \eta + \sum_{i=1}^{n} (y_i - y_{i-1}) \ln \left( a^{b^i} - a^{b^{i-1}} \right) \\
- \sum_{i=1}^{n} \ln (y_i - y_{i-1}) - \eta \sum_{i=1}^{n} \left( a^{b^i} - a^{b^{i-1}} \right) \\
- \frac{\partial \ell}{\partial \eta} = \frac{y_n}{\eta} - a^{b^{n-1}} - 1 = 0 . \\
\eta = \frac{y_n}{a^{b^{n-1}} - 1} ; \quad \ln b > 0 . \]  

(4.154)

The derivative of \( \ell_8 \) with respect to \( a \) is
\[ \frac{\partial \ell_8}{\partial a} = \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) \frac{\left( b_i^a b_i^{h-1} - b_{i-1}^a b_{i-1}^{h-1} \right)}{\left( a^a b_i^{h-1} - a^a b_{i-1}^{h-1} \right)} \]

The summation appearing in the second term of the last equation is equal to

\[ b_{n-1}^a b_n^{a-1} - 1. \]

Then, after equating \( \frac{\partial \ell_8}{\partial a} \) to zero, we have

\[ \frac{y_n}{a} \left[ b_n^a b_n^{a-1} - b_{n-1}^a b_{n-1}^{a-1} \right] = \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) \frac{\left( b_i^a b_i^{h-1} - b_{i-1}^a b_{i-1}^{h-1} \right)}{\left( a^a b_i^{h-1} - a^a b_{i-1}^{h-1} \right)}. \]

Differentiate (4.153) with respect to \( b \) and equate to zero. Thus, the following equation results

\[ \frac{\partial \ell_8}{\partial b} = \left( \ln a \right) \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) \frac{\left( b_i^a b_i^{h-1} - b_{i-1}^a b_{i-1}^{h-1} \right)}{\left( a^a b_i^{h-1} - a^a b_{i-1}^{h-1} \right)}, \]

which can be rewritten as

\[ \frac{y_n b_n^{a-1} b_n^{h-1}}{a^{b_n^{a-1} b_n^{h-1}}} = \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) \frac{\left( b_i^a b_i^{h-1} - b_{i-1}^a b_{i-1}^{h-1} \right)}{\left( a^a b_i^{h-1} - a^a b_{i-1}^{h-1} \right)}. \]

### 4.4.1.9 The logistic model

The form of this model we will use in this Subsection is indicated in equation (4.87). The logarithm of the likelihood function is
$$\ell_n = \sum_{i=1}^{n} \left[ (y_i - y_{i-1}) \ln \eta + (y_i - y_{i-1}) \ln \frac{m e^{-pt_i -1} - e^{-pt_i}}{1 + me^{-pt_i -1}} - \ln(y_i - y_{i-1}) \right]$$

$$- \eta m \sum_{i=1}^{n} \left[ \frac{e^{-pt_i -1} - e^{-pt_i}}{1 + me^{-pt_i -1}} \right].$$

(4.157)

Differentiate with respect to $\eta$ and equate to zero. Then, after some simple calculations, obtain

$$\eta = \frac{y_n (1 + m)(1 + me^{-pt})}{m^2(1 - e^{-pt_n})}. \quad (4.158)$$

Again differentiate with respect to $m$ to get

$$\frac{\partial \ell_n}{\partial m} = \frac{y_n}{m} - \sum_{i=1}^{n} (y_i - y_{i-1}) \left( \frac{e^{-pt_i}}{1 + me^{-pt_i}} + \frac{e^{-pt_i -1}}{1 + me^{-pt_i -1}} \right) - \frac{y_n (1 - m^2 e^{-pt})}{m(1 + m)(1 + me^{-pt_n})}.$$

Equating to zero, we have the second likelihood equation in the form

$$\frac{y_n}{m} \left( \frac{1 - m^2 e^{-pt_n}}{(1 + m)(1 + me^{-pt_n})} \right) = \sum_{i=1}^{n} (y_i - y_{i-1}) \left( \frac{e^{-pt_i}}{1 + me^{-pt_i}} + \frac{e^{-pt_i -1}}{1 + me^{-pt_i -1}} \right). \quad (4.159)$$

To obtain the third equation, differentiate with respect to $p$, namely

$$\frac{\partial \ell_n}{\partial p} = \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) \left( t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}} \right) + \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) m t_i e^{-pt_i}$$

$$+ \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) m t_{i-1} e^{-pt_{i-1}} - \frac{y_n (1 + m) e^{-pt_n}}{(1 + me^{-pt_n}) (1 - e^{-pt_n})}.$$

Equating to zero, we get

$$\frac{y_n (1 + m) t_n e^{-pt_n}}{(1 + me^{-pt_n}) (1 - e^{-pt_n})} = \frac{\sum_{i=1}^{n} \left( y_i - y_{i-1} \right)}{1 + me^{-pt_i} + t_i e^{-pt_i} - t_{i-1} e^{-pt_{i-1}}}.$$

(4.160)
All the likelihood equations in this section are shown in Table 4.2.

<table>
<thead>
<tr>
<th>Model</th>
<th>Likelihood Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>K-stage Erlangian (gamma) growth curve model</strong></td>
<td>[ a = y_n \left[ 1 - e^{-bt} \sum_{j=0}^{K-1} \left( \frac{bt_n}{j!} \right)^j \right] ]</td>
</tr>
<tr>
<td>[ y(t) = a \left[ 1 - e^{-bt} \sum_{j=0}^{K-1} \left( \frac{bt_n}{j!} \right)^j \right] ]</td>
<td>[ \frac{y_n t_n (bt_n)^{K-1} e^{-bt_n}}{(K-1)! - e^{-bt} \sum_{j=0}^{K-1} \left( \frac{bt_n}{j!} \right)^j} ]</td>
</tr>
<tr>
<td>[ r(t) = a \left[ \sum_{j=1}^{K} \left( \frac{bt_n}{j!} \right)^j - e^{-bt} \sum_{j=0}^{K-1} \left( \frac{bt_n}{j!} \right)^j \right] ]</td>
<td>[ \frac{y_n t_n \left( \frac{bt_n}{(K-1)!} \right)^{K-1} e^{-bt_n}}{1 - e^{-bt} \sum_{j=0}^{K-1} \left( \frac{bt_n}{j!} \right)^j} ]</td>
</tr>
<tr>
<td><strong>exponential SRGM</strong></td>
<td>[ a = \frac{y_n}{1 - e^{-bt_n}} ]</td>
</tr>
<tr>
<td>[ y_1(t) = a \left[ 1 - e^{-bt} \right] ]</td>
<td>[ \frac{y_n t_n e^{-bt_n}}{(1 - e^{-bt_n})} = \sum_{i=1}^{\infty} \left( \frac{y_{1i} - y_{1i+1}}{e^{-bt_{1i+1}} - e^{-bt_{1i}}} \right) ]</td>
</tr>
<tr>
<td><strong>delayed S-shaped SRGM</strong></td>
<td>[ a = \frac{y_n}{1 - (1 + bt_n) e^{-bt_n}} ]</td>
</tr>
<tr>
<td>[ y_2(t) = a \left[ 1 - (1 + bt)e^{-bt} \right] ]</td>
<td>[ \frac{y_n t_n^2 e^{-bt_n}}{1 - (1 + bt_n) e^{-bt_n}} = \sum_{i=1}^{\infty} \left( \frac{y_{2i} - y_{2i+1}}{(1 + bt_{2i+1}) e^{-bt_{2i+1}} - (1 + bt_{2i}) e^{-bt_{2i}}} \right) ]</td>
</tr>
<tr>
<td>The K-model at K = 3</td>
<td>[ a = \frac{y_n}{1 - \left( 1 + bt_n + \frac{1}{2} (bt_n)^2 \right) e^{-bt_n}} ]</td>
</tr>
<tr>
<td>[ y_3(t) = a \left[ 1 - (1 + \frac{1}{2} y)e^{-bt} \right] ]</td>
<td>[ \frac{y_n t_n^2 e^{-bt_n}}{1 - \left( 1 + bt_n + \frac{1}{2} (bt_n)^2 \right) e^{-bt_n}} ]</td>
</tr>
<tr>
<td>, where [ y = bt(2 + bt) ]</td>
<td>[ \sum_{i=1}^{\infty} \left( \frac{y_{3i} - y_{3i+1}}{(1 + bt_{3i+1}) e^{-bt_{3i+1}} - (1 + bt_{3i}) e^{-bt_{3i}}} \right) ]</td>
</tr>
</tbody>
</table>

Table 4.2: Likelihood equations for error-detection count data.
<table>
<thead>
<tr>
<th>Model</th>
<th>Likelihood Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>The K-model at K = 4</td>
<td>( y_n = \frac{a}{1 - \left(1 + bt_n + \frac{1}{2}(bt_n)^2 + \frac{1}{6}(bt_n)^3\right)e^{-bt_n}} )</td>
</tr>
<tr>
<td>( y_4(t) = a \left[1 - \left(1 + \frac{1}{6}y_1\right)e^{-bt}\right] )</td>
<td>( \sum_{i=1}^{n} \left(1 + bt_i + \frac{1}{2}(bt_i)^2 + \frac{1}{6}(bt_i)^3\right)e^{-bt_i} - \left(1 + bt_i + \frac{1}{2}(bt_i)^2 + \frac{1}{6}(bt_i)^3\right)e^{-bt_i} )</td>
</tr>
<tr>
<td>modified exponential SRGM</td>
<td>( a = \frac{\sum_{j=1}^{n} p_j \left(1 - e^{-bt_n}\right)}{\sum_{j=1}^{n} p_j \left(1 - e^{-bt_n}\right)} )</td>
</tr>
<tr>
<td>( m_p(t) = a \sum_{j=1}^{\infty} p_j \left(1 - e^{-bt_n}\right) )</td>
<td>( \frac{\sum_{j=1}^{n} p_j \left(1 - e^{-bt_n}\right)}{\sum_{j=1}^{n} p_j \left(1 - e^{-bt_n}\right)} = \sum_{i=1}^{n} \frac{(y_i - y_{i-1})\sum_{j=1}^{n} p_j \left(t_{i+1} - t_i\right)e^{-bt_i}}{2 \sum_{j=1}^{n} p_j \left(1 - e^{-bt_n}\right)} )</td>
</tr>
<tr>
<td>inflection S-shaped SRGM</td>
<td>( a = \frac{\sum_{j=1}^{n} p_j \left(1 - e^{-bt_n}\right)}{\sum_{j=1}^{n} p_j \left(1 - e^{-bt_n}\right)} )</td>
</tr>
<tr>
<td>( f(t) = \frac{\left(1 - e^{-bt}\right)}{\left(1 + ce^{-bt}\right)} )</td>
<td>( \frac{(1 + ce^{-bt})e^{bt_n}}{\left(1 - e^{-bt}\right)} )</td>
</tr>
<tr>
<td>( \sum_{i=1}^{n} (y_i - y_{i-1}) \left(1 + ce^{-bt_i}\right)e^{-bt_i} )</td>
<td>( \sum_{i=1}^{n} (y_i - y_{i-1}) \left(1 + ce^{-bt_i}\right)e^{-bt_i} )</td>
</tr>
<tr>
<td>testing-effort dependent SRGM</td>
<td>( a = \frac{\sum_{j=1}^{n} p_j \left(1 - e^{-bt_n}\right)}{\sum_{j=1}^{n} p_j \left(1 - e^{-bt_n}\right)} )</td>
</tr>
<tr>
<td>( T(t) = a \left[1 - \exp\left(-r\left(1 - e^{-bt}\right)\right)\right] )</td>
<td>( \frac{\sum_{i=1}^{n} (y_i - y_{i-1}) \left(W(t_i) - W(t_{i-1})\right)}{\sum_{i=1}^{n} (y_i - y_{i-1}) \left(\exp[-rW(t_i)] - \exp[-rW(t_{i-1})]\right)} )</td>
</tr>
<tr>
<td>Littlewood NHPP model</td>
<td>( a = \frac{y_n}{\left(b + t_n\right)^{1+b}} )</td>
</tr>
<tr>
<td>( L(t) = a \left(1 - b^c(b + t)^{-c}\right) )</td>
<td>( \left(1 + \frac{1}{(b + t_n)^{1+b}} - b^c(b + t_n)^{-c}\right)\sum_{i=1}^{n} \left(\frac{(y_i - y_{i-1})}{(b + t_{i+1})^{1+b} - (b + t_i)^{1+b}}\right) )</td>
</tr>
</tbody>
</table>

Table 4.2: Continued.
<table>
<thead>
<tr>
<th>Model</th>
<th>Likelihood Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Littlewood NHPP model</td>
<td>( L_t(t) = a \left( 1 - b^e(b+t)^b \right) )</td>
</tr>
<tr>
<td></td>
<td>( y_n \left( \frac{\ln(b+t_n) - \ln b}{(b+t_n)^b - b^e(b+t)^b} \right) = )</td>
</tr>
<tr>
<td></td>
<td>( \sum_{i=1}^{n} \left( \frac{y_i - y_{i-1}}{b_{i-1}^e(b+t_{i-1})} \right) )</td>
</tr>
<tr>
<td>Duane model (continued)</td>
<td>( D(t) = at^b )</td>
</tr>
<tr>
<td></td>
<td>( y_n \ln t_n = \sum_{i=1}^{n} \left( \frac{y_i - y_{i-1}}{b_{i-1}^e(b+t_{i-1})} \right) )</td>
</tr>
<tr>
<td>Gompertz SRGM</td>
<td>( G_p(t) = \eta a^b )</td>
</tr>
<tr>
<td></td>
<td>( \eta = \frac{y_n}{a(b_{b-1}^e - 1)} )</td>
</tr>
<tr>
<td></td>
<td>( \sum_{i=1}^{n} \left( \frac{y_i - y_{i-1}}{a(b_{i-1}^e - 1)} \right) )</td>
</tr>
<tr>
<td></td>
<td>( y_n a^b - b_{b-1}^e = )</td>
</tr>
<tr>
<td></td>
<td>( \sum_{i=1}^{n} \left( \frac{y_i - y_{i-1}}{a(b_{i-1}^e - 1)} \right) )</td>
</tr>
<tr>
<td>Logistic SRGM</td>
<td>( L_0(t) = \frac{\eta}{1 + me^{-\beta t}} )</td>
</tr>
<tr>
<td></td>
<td>( \eta = \frac{y_n(1 + m)(1 + me^{-\beta t})}{m^2(1 - e^{-\beta t})} )</td>
</tr>
<tr>
<td></td>
<td>( \sum_{i=1}^{n} \left( \frac{y_i - y_{i-1}}{1 + me^{-\beta t}} \right) )</td>
</tr>
<tr>
<td></td>
<td>( \frac{y_n(1 + m)ke^{-\beta t_n}}{(1 + me^{-\beta t_n})} = )</td>
</tr>
<tr>
<td></td>
<td>( \sum_{i=1}^{n} \left( \frac{y_i - y_{i-1}}{1 + me^{-\beta t}} \right) )</td>
</tr>
</tbody>
</table>

Table 4.2: Continued.

### 4.4.2 Estimation of the SRGMs Parameters in case of Time Data

Assessment of the reliability of the software at any time during the test process depends on the model assumed for the activation of faulty systems. Most of the models in the literature...
pertaining to this matter are the so called “time domain” models which consider the time till a fault is activated as a random variable realized according to some stochastic process. This last is generally taken as a non-homogeneous Poisson process. In this situation the collected data are called failure-occurrence time data. In this Section we will introduce the parameter’s estimators by using the MLE method, considering this case with the models mentioned in Section 3.3.

Suppose that the failure-occurrence time data \( s_i \) (\( i = 1, 2, \ldots, n \)) are observed during the testing phase. Then, the joint density function of the observed data, that is, the likelihood function for estimating the unknown parameters in an NHPP model with the mean value function \( H(t) \), is given by

\[
L(a, \theta | S) = \exp\{-H(s_n)\} \prod_{i=1}^{n} h(s_i) ,
\]

where

\[
h(s_i) = \left. \frac{dH(t)}{dt} \right|_{t=s_i} ,
\]

and \( a \) is the expected number of errors in the system.

\[
H(s_n) = H(t)|_{t=s_n} .
\]

### 4.4.2.1 Estimation of the parameters of the K-stage Erlangian model

Consider the mean value function of this model given by (4.53). Then set \( s = s_n \) to obtain

\[
\gamma(s_n) = a \left\{ 1 - e^{-bs_n} \sum_{j=0}^{K-1} \left( \frac{bs_n}{j} \right)^j \right\} ,
\]

where \( b \) is the error detection rate in the system. The derivative of this function at \( s = s_i \) is
\[
\frac{d\gamma(t)}{dt} \bigg|_{t=s_i} = abe^{-bs_i} \left[ \sum_{j=0}^{K-1} \frac{(bs_i)^j}{j!} \right] - \sum_{j=1}^{K-1} \frac{(bs_i)^{j-1}}{(j-1)!} \\
= abe^{-bs_i} \frac{(bs_i)^{K-1}}{(K-1)!}.
\] (4.164)

The likelihood function is

\[ L_1(a, b | \mathcal{S}) = (ab)^n \exp \left[ -a \left( 1 - e^{-bs_a} \sum_{j=0}^{K-1} \frac{(bs_a)^j}{j!} \right) - b \sum_{i=1}^{n} s_i \right] \prod_{i=1}^{n} \frac{(bs_i)^{K-1}}{(K-1)!}. \] (4.165)

Taking the natural logarithm of both sides, we have

\[ \eta_1 = n \ln a + n \ln b - a \left( 1 - e^{-bs_a} \sum_{j=0}^{K-1} \frac{(bs_a)^j}{j!} \right) - b \sum_{i=1}^{n} s_i + \sum_{i=1}^{n} \ln \frac{(bs_i)^{K-1}}{(K-1)!}. \] (4.166)

Proceed to get the MLEs of \( a, b \). That is, differentiate \( \eta_1 \) with respect to each of \( a \) and \( b \), namely

\[ \frac{\partial \eta_1}{\partial a} = - \left( 1 - e^{-bs_a} \sum_{j=0}^{K-1} \frac{(bs_a)^j}{j!} \right) + \frac{n}{a} = 0, \]
\[ \frac{\partial \eta_1}{\partial b} = -a e^{-bs_a} \frac{(bs_a)^{K-1}}{(K-1)!} - \sum_{i=1}^{n} s_i + \frac{nK}{b} = 0. \]

The two equations above can be rewritten as follows

\[ a = n \left\{ 1 - e^{-bs_a} \sum_{j=0}^{K-1} \frac{(bs_a)^j}{j!} \right\}, \] (4.167)
\[ \frac{nK}{b} = \sum_{i=1}^{n} s_i + nK \frac{(bs_a)^{K-1}}{(K-1)!} \left\{ 1 - e^{-bs_a} \sum_{j=0}^{K-1} \frac{(bs_a)^j}{j!} \right\}. \] (4.168)
As we mentioned in Section 4.3, the values \( K = 1 \) and \( K = 2 \) correspond to the cases of the exponential model and the S-shaped model respectively. For these two cases and also for the cases \( K = 3 \) and \( 4 \) the MLEs of \( a \) and \( b \) are mentioned in Table 4.3.

### 4.4.2.2 Estimation of the parameters of the modified exponential model

For this model the mean value function is evaluated at \( s = s_\text{n} \) (according to equation (4.24))

\[
m_p(s_\text{n}) = a \sum_{j=1}^{2} p_j \left(1 - e^{-b_j s_\text{n}}\right),
\]

where \( 0 < b_2 < b_1 < 1 \), \( \sum_{i=1}^{2} p_i = 1 \), \( 0 < p_j < 1 \), \( j = 1, 2 \).

Also, its derivative at \( s = s_\text{i} \) is

\[
g_2(s_\text{i}) = \frac{d m(t)}{dt} \bigg|_{t=s_\text{i}} = a \sum_{j=1}^{2} p_j b_j e^{-b_j s_\text{i}}.
\]

Therefore the likelihood function is given in the form

\[
L_2(a, \theta | S) = \exp \left\{ -a \sum_{j=1}^{2} p_j \left(1 - e^{-b_j s_\text{n}}\right) \prod_{i=1}^{n} a \sum_{j=1}^{2} p_j b_j e^{-b_j s_\text{i}} \right\}.
\]

Take the natural logarithm of both sides and proceed to get the estimates of \( a \) and \( b_j \) for \( j = 1, 2 \). Then, the likelihood function and its derivatives are

\[
\eta_2 = -a \sum_{j=1}^{2} p_j \left(1 - e^{-b_j s_\text{n}}\right) + \sum_{i=1}^{n} \ln \left( a \sum_{j=1}^{2} p_j b_j e^{-b_j s_\text{i}} \right),
\]

\[
\frac{\partial \eta_2}{\partial a} = -\sum_{j=1}^{2} p_j \left(1 - e^{-b_j s_\text{n}}\right) + \frac{n}{a} = 0,
\]

106
The last two equations are rearranged as

\[ a = n \sqrt{\left\{ \sum_{j=1}^{2} p_j \left(1 - e^{-b_j s_n}\right) \right\}}, \quad (4.172) \]

\[ \frac{\text{ns}_n \sum_{j=1}^{2} e^{-b_j s_n}}{\sum_{j=1}^{2} p_j (1 - e^{-b_j s_n})} = \frac{\sum_{j=1}^{n} \sum_{j=1}^{2} p_j e^{-b_j s_j} (1 - s_j b_j)}{\sum_{j=1}^{2} p_j (1 - e^{-b_j s_n})}. \quad (4.173) \]

**4.4.2.3 Estimation of the parameters of the inflection S-shaped SRGM**

Apply (4.161) to the model given by (4.37), by first obtaining

\[ I(s_n) = a \left(1 - e^{-b_s s_n}\right) \left(1 + ce^{-b_s s_n}\right), \quad (4.174) \]

and

\[ g_3(s_i) = \frac{dI(t)}{dt} \bigg|_{t=s_i} = \frac{abe^{-b_s s_i} (1 + c)}{(1 + ce^{-b_s s_i})^2}. \quad (4.175) \]

From (4.174) and (4.175) the likelihood function is

\[ L_3(a, \theta | S) = \exp\left\{ -a \left(1 - e^{-b_s s_n}\right) \left(1 + ce^{-b_s s_n}\right) \prod_{i=1}^{n} \left\{ abe^{-b_s s_i} (1 + c) \left(1 + ce^{-b_s s_i}\right)^2 \right\} \right\}. \quad (4.176) \]

Using the usual procedures to get the estimates of \( a \) and \( b \), we get
\[ \eta_3 = \ln(a, b \mid S) = \frac{-a(1 - e^{-bs_n})}{1 + ce^{-bs_n}} + \sum_{i=1}^{n} \ln \left[ \left\{ b e^{-(b+1)(1+c)} \right\}/\left\{ 1 + ce^{-bs_i} \right\} \right]. \]

\[
\frac{\partial \eta_3}{\partial a} = \frac{-(1 - e^{-bs_n})}{1 + ce^{-bs_n}} + \frac{n}{a} = 0.
\]

The last equation can be rewritten as
\[ a = \frac{n(1 + ce^{-bs_n})}{1 - e^{-bs_n}}. \quad (4.177)\]

After some manipulation the equation \( \frac{\partial \eta_3}{\partial b} \) can be reduced to
\[
\frac{na_n e^{-bs_n}(1+c)}{(1-e^{-bs_n})(1+ce^{-bs_n})} = n b - \sum_{i=1}^{n} s_i + 2 \sum_{i=1}^{n} cs_i e^{-bs_i}. \quad (4.178)
\]

### 4.4.2.4 Estimation of the testing-effort dependent SRGM parameters

Consider the model given by (4.45), where
\[ W(t) = \alpha \left(1 - e^{-\beta m} \right). \]

Then for this model the mean value function evaluated at \( t = s_n \) is
\[
T(s_n) = a \left[ 1 - \exp \left\{ -\alpha \left(1 - e^{-bs_n} \right) \right\} \right], \quad (4.179)
\]

and from equation (4.179) we have
\[
\tau(s_i) = a r \alpha \beta ms_i^{m-1} \exp \left\{ -\beta s_i^m \right\} \exp \left\{ -\alpha \left(1 - e^{-bs_i^m} \right) \right\}. \quad (4.180)
\]

Hence, the likelihood function is
\[ L_4(a, \theta | S) = a^n r^n \beta^n m^n \exp \{- \beta s_i^m \} \exp \left[ -a \left\{ 1 - \exp \left\{ -r \alpha \left( 1 - e^{-\beta s_i^m} \right) \right\} \right\} \right] \]

\[
\times \prod_{i=1}^{n} \exp \left\{ -r \alpha \sum_{i=1}^{n} \left( 1 - e^{-\beta s_i^m} \right) \right\} ,
\]

where \( \theta = r. \)

Proceed to get the estimates of \( a \) and \( r. \) We have

\[ \eta_4 = \ln L_4(a, \theta | S) = n \ln a + n \ln r + n \ln \beta m - a \left\{ 1 - \exp \left\{ -r \alpha \left( 1 - e^{-\beta s_i^m} \right) \right\} \right\} \]

\[
+ \sum_{i=1}^{n} (m-1) \ln s_i - r \alpha \sum_{i=1}^{n} \left( 1 - e^{-\beta s_i^m} \right) ,
\]

\[ (4.182) \]

\[
\frac{\partial \eta_4}{\partial a} = \left[ 1 - \exp \left\{ -r \alpha \left( 1 - e^{-\beta s_i^m} \right) \right\} \right] + \frac{n}{a} = 0 ,
\]

\[
\frac{\partial \eta_4}{\partial r} = -a \alpha n \left( 1 - e^{-\beta s_i^m} \right) \exp \left\{ -r \alpha \left( 1 - e^{-\beta s_i^m} \right) \right\} + \frac{n}{r} - r \alpha \sum_{i=1}^{n} \left( 1 - e^{-\beta s_i^m} \right) = 0 .
\]

The last two equations are equivalent to

\[ a = n / \left\{ 1 - \exp \left\{ -r \alpha \left( 1 - e^{-\beta s_i^m} \right) \right\} \right\} . \]

\[ (4.183) \]

\[ \frac{n}{r} = n \alpha \left( 1 - e^{-\beta s_i^m} \right) \exp \left\{ -r \alpha \left( 1 - e^{-\beta s_i^m} \right) \right\} + \alpha \sum_{i=1}^{n} \left( 1 - e^{-\beta s_i^m} \right) . \]

\[ (4.184) \]

Solving (4.183) and (4.184) for \( a \) and \( r, \) we get the required estimates.

**4.4.2.5 Estimation of Littlewood NHPP model's parameters**

Returning to this model which is given by (4.67), we have
Applying for the likelihood function, we have

\[ L_s(a, b | S) = a^n b^m c^n \exp \left[ -a \left( 1 - b^c (b + s_n)^{-c} \right) \right] \prod_{i=1}^{n} (b + s_i)^{-c} . \]

Put

\[ \eta_s = \ln L_s(a, b | S) , \]

and differentiate with respect to \( a \) and equate to zero to obtain

\[ \frac{\partial \eta_s}{\partial a} = -\left( 1 - b^c (b + s_n)^{-c} \right) + \frac{n}{a} = 0 , \]

or equivalently

\[ a = n \left/ \left( 1 - b^c (b + s_n)^{-c} \right) \right. . \]  (4.186)

Similarly, differentiate with respect to \( b \) then \( c \), to obtain

\[ \frac{\partial \eta_s}{\partial b} = ab^c (b + s_n)^{-c} \ln \left( b/(b + s_n) \right) + n \ln b + \frac{n}{c} - \sum_{i=1}^{n} \ln(b + s_i) , \]

and therefore

\[ \frac{nc}{b} = (c + 1) \sum_{i=1}^{n} \frac{1}{b + s_i} - \frac{ncs_n b^{-c-1} (b + s_n)^{-c-1}}{1 - b^c (b + s_n)^{-c}} . \]  (4.187)

\[ \frac{n}{c} = \sum_{i=1}^{n} \ln(b + s_i) - ab^c (b + s_n)^{-c} \ln \left( b/(b + s_n) \right) - n \ln b . \]  (4.188)
The estimates \( \hat{a}, \hat{b} \) and \( \hat{c} \) could be evaluated by solving (4.186), (4.187) and (4.188) iteratively.

### 4.4.2.6 Estimation of Duane model’s parameters

It is easy to work out for this model, where

\[
D(s_n) = a s_n^b, \\
\frac{d(s_i)}{d} = a b s_i^{b-1}
\]

Then

\[
L_6(a, b | S) = e^{-a s_n^b} a^n b^n \prod_{i=1}^{n} s_i^{b-1} ,
\]

and

\[
\eta_6 = \ln L_6(a, b | S) \\
= -a s_n^b + n \ln a + n \ln b + \sum_{i=1}^{n} (b - 1) \ln s_i .
\]

Proceed to estimate \( a \) and \( b \). Then

\[
\frac{\partial \eta_6}{\partial a} = -s_n^b + \frac{n}{a} = 0 .
\]

Solving for \( a \), we have

\[
a = \frac{n}{s_n^b} .
\]

Also

\[
\frac{\partial \eta_6}{\partial b} = 0 ,
\]

giving

\[
\frac{n}{b} = n \ln s_n - \sum_{i=1}^{n} \ln s_i .
\]
4.4.2.7 Estimation of Gompertz and logistic growth models’ parameters

For the Gompertz model we have by equation (4.93),

\[
G_p(s_n) = \eta a^b s_n, \quad g_p(s_i) = \eta a^b b^s_i \ln(a) \ln(b)
\]

where

\[
\eta > 0, \quad 0 < a < 1, \quad 0 < b < 1.
\]

Hence the log likelihood function is

\[
L_p(a, b, \eta | S) = \exp\left(-\eta a^b \right) \cdot \eta^n (\ln a)^a (\ln b)^b \prod_{i=1}^{n} a^b b^s_i.
\]  

Differentiate with respect to \( \eta \), \( a \) and \( b \) and equate to zero to get the estimates of \( \eta \), \( a \) and \( b \) respectively. These give

\[
\eta = \frac{n}{a^b s_n},
\]

\[
\frac{nb^a}{a} = \frac{1}{a} \left( \frac{n}{\ln a} + \sum_{i=1}^{n} b^s_i \right),
\]

and

\[
n b^a - 1 \ln a = \frac{1}{b} \left( \frac{n}{\ln b} + \sum_{i=1}^{n} s_i b^s_i \ln a \right).
\]

For the logistic growth model we have, from (4.87),

\[
L_s(s_n) = \frac{\eta}{1 + me^{-pn}}, \quad \ell_s(s_i) = \frac{\eta pm e^{-ps_i}}{(1 + me^{-pn})^2}
\]

where \( \eta > 0, \ m > 0, \ p > 0 \).
The likelihood function is

\[ L_g(T_i, p, m | s) = \exp\left( -\eta \frac{r_i - m e^{-p s_i}}{1 + m e^{-p s_i}} \right) \prod_{i=1}^{n} \frac{\eta m e^{-p s_i}}{(1 + m e^{-p s_i})^2} . \]  (4.200)

Proceed as before, we obtain the estimates of \( \eta, p, \) and \( m \) respectively as follows

\[ \ell_g = \ln L_g(\eta, p, m | s) = -\frac{\eta}{1 + m e^{-p s_i}} + n \ln \eta + n \ln p + n \ln m - \sum_{i=1}^{n} ps_i - 2 \sum_{i=1}^{n} \ln(1 + m e^{-p s_i}) . \]  (4.201)

where \( \eta, p \) and \( m \) have to be estimated. Proceeding for this purpose, we get

\[ \eta = n \left( 1 + m e^{-p s_i} \right) . \]  (4.202)

\[ \frac{n}{p} = \frac{n}{\sum_{i=1}^{n} s_i} + m \left\{ \frac{n s_n e^{-p s_n}}{1 + m e^{-p s_n}} - \frac{2 n}{\sum_{i=1}^{n} s_i e^{-p s_i}} \right\} , \]  (4.203)

and

\[ \frac{n}{m} = 2 \frac{n}{\sum_{i=1}^{n} e^{-p s_i}} + \frac{n e^{-p s_n}}{1 + m e^{-p s_n}} . \]  (4.204)

Equations (4.202), (4.203) and (4.204) can be solved numerically to obtain the MLEs \( \hat{\eta}, \hat{a}, \) and \( \hat{p} \). All the likelihood equations in this Section are shown in Table 4.3.
<table>
<thead>
<tr>
<th>Model</th>
<th>Likelihood Equations</th>
</tr>
</thead>
</table>
| K-stage Erlangian (gamma) growth curve model | \[ \gamma(t) = a \left( 1 - e^{-bs_n^K \sum_{j=0}^{K-1} \frac{(bt)^j}{j!}} \right) \]  
\[ a = n \left( 1 - e^{-bs_n^K \sum_{j=0}^{K-1} \frac{(bt)^j}{j!}} \right) \] |
| exponential SRGM | \[ \gamma_1(t) = a \left( 1 - e^{-bt} \right) \]  
\[ a = \frac{n}{1 - e^{-bs_n^a}} \] |
| delayed S-shaped SRGM | \[ \gamma_2(t) = a \left[ 1 - (1 + bt)e^{-bt} \right] \]  
\[ a = \frac{n}{1 - (1 + bs_n^a)} \]  
\[ b = \frac{n}{\sum_{i=1}^{n} s_i + \frac{ns_ne^{-bs_n^a}}{1 - e^{-bs_n^a}}} \] |
| The K-model at K = 3 | \[ \gamma_3(t) = a \left[ 1 - \left( 1 + \frac{1}{2}y \right)e^{-bt} \right] \]  
\[ a = \frac{n}{1 - \left( 1 + bs_n^a + \frac{1}{2}(bs_n^a)^2 \right)e^{-bs_n^a}} \]  
\[ y = bt(2 + bt) \] |
| The K-model at K = 4 | \[ \gamma_4(t) = a \left[ 1 - \left( 1 + \frac{1}{6}y_1 \right)e^{-bt} \right] \]  
\[ a = \frac{n}{1 - \left( 1 + bs_n^a + \frac{1}{2}(bs_n^a)^2 + \frac{1}{6}(bs_n^a)^3 \right)e^{-bs_n^a}} \]  
\[ y_1 = bt(6 + 3bt + (bt)^2) \] |
| modified exponential SRGM | \[ m_p(t) = a \sum_{j=1}^{2} p_j \left( 1 - e^{-b_j t} \right) \]  
\[ a = b \sum_{j=1}^{2} p_j \left( 1 - e^{-b_j p_n} \right) \]  
\[ ns_n^2 \sum_{j=1}^{2} p_j e^{-b_j s_n^a} = \sum_{j=1}^{2} \frac{2 p_j e^{-b_j p_f}}{\sum_{j=1}^{2} p_j \left( 1 - e^{-b_j s_n^a} \right)} \]  
\[ \sum_{j=1}^{2} \frac{2 p_j b_j e^{-b_j p_f}}{\sum_{j=1}^{2} p_j \left( 1 - e^{-b_j s_n^a} \right)} \] |

Table 4.3: Likelihood equations for failure-occurrence time data.
<table>
<thead>
<tr>
<th>Model</th>
<th>Likelihood Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>inflection S-shaped SRGM</td>
<td>$I(t) = \frac{a(t - e^{-ct})}{1 + ce^{-ct}}$</td>
</tr>
<tr>
<td></td>
<td>$a = \frac{n(1 + ce^{-bs_n})}{1 - e^{-bs_n}}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{ns_n e^{-bs_n}}{(1 + ce^{-bs_n})} (1 - e^{-bs_n}) = \frac{n}{b} - \sum_{i=1}^{n} s_i + 2\sum_{i=1}^{n} e^{-bs_i}$</td>
</tr>
<tr>
<td>testing-effort dependent SRGM</td>
<td>$T(t) = a \left[1 - \exp\left(-r_0(1 - e^{-pt})\right)\right]$</td>
</tr>
<tr>
<td></td>
<td>$a = \frac{n}{1 - \exp\left(-r_0(1 - e^{-pt})\right)}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{n}{r} = \frac{\alpha}{1 - \exp\left(-r_0(1 - e^{-pt})\right)} + \alpha \sum_{i=1}^{n} (1 - e^{-pt})$</td>
</tr>
<tr>
<td>Littlewood NHPP model</td>
<td>$I_t(t) = a \left[1 - b^c(b + t)^{-c}\right]$</td>
</tr>
<tr>
<td></td>
<td>$a = \frac{n}{1 - b^c(b + s_n)^{-c}}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{nc}{b} = (c + 1) \sum_{i=1}^{n} \frac{1}{b + s_i} - \frac{nc_n b^{-c-1} (b + s_n)^{-c-1}}{1 - b^c(b + s_n)^{-c}}$</td>
</tr>
<tr>
<td></td>
<td>$n = \sum_{i=1}^{n} \ln(b + s_i) - ab^c(b + s_n)^{-c} \ln\left{\frac{b}{(b + s_n)}\right} - n \ln b$</td>
</tr>
<tr>
<td>Duane model</td>
<td>$D(t) = at^b$</td>
</tr>
<tr>
<td></td>
<td>$a = \frac{n}{s_n b^b}$</td>
</tr>
<tr>
<td></td>
<td>$\ln s_n = \frac{n}{b} + \sum_{i=1}^{n} s_i$</td>
</tr>
<tr>
<td>Gompertz SRGM</td>
<td>$G_p(t) = \eta a^b$</td>
</tr>
<tr>
<td></td>
<td>$\eta = \frac{n}{a b^a}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{nb^{s_n}}{a} = \left(\frac{n}{a} + \sum_{i=1}^{n} b^{s_i}\right)$</td>
</tr>
<tr>
<td></td>
<td>$ns_n b^{s_n-1} \ln a = \left(\frac{n}{b} + \sum_{i=1}^{n} s_i + \sum_{i=1}^{n} s_i b^{s_i} \ln a\right)$</td>
</tr>
<tr>
<td>logistic SRGM</td>
<td>$L_0(t) = \frac{\eta}{1 + me^{-pt}}$</td>
</tr>
<tr>
<td></td>
<td>$\eta = n(1 + me^{-ps_n})$</td>
</tr>
<tr>
<td></td>
<td>$\frac{n}{p} = \sum_{i=1}^{n} s_i + \frac{ns_n e^{-p s_n} - 2\sum_{i=1}^{n} s_i e^{-p s_i}}{1 + me^{-p s_n}}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{n}{m} = 2\sum_{i=1}^{n} e^{-p s_i} + \frac{me^{-p s_n}}{1 + me^{-p s_n}}$</td>
</tr>
</tbody>
</table>

Table 4.3: Continued.
4.5 Data analysis

In this section, we analyze four software failure data sets to provide empirical evaluation of the model comparison techniques using eleven NHPP models. We developed a set of software tools in Java in order to perform the evaluations. In more detail, our software tools help to perform the following functions:

- Derive the estimates of the unknown parameters that are involved in all the used models. This is carried out by using the maximum likelihood estimation method;
- Evaluate the reliability prediction results for all the used models;
- Use four goodness-of-fit tests to compare the models we investigate.

We also used the MATLAB software to produce graphs for:

- The reliability prediction results;
- The estimated expected number of remaining faults.

4.5.1 Data tables

We chose four data sets for our evaluations. These are presented in Tables 4.4 – 4.7. The data set shown in Table 4.4 is from Goel and Okumoto (1979), which originated from the U.S. Navy Fleet Computer Programming Center. These failure data were collected during the development phase of the software for the real-time multicomputer complex system that is the central part of the Navel Tactical Data System (NTDS). The data in Table 4.5 were presented by Moranda (1975a) and pertain to a record of errors which occurred during the debugging of a data reduction program called the F11-D program. This program consists of “approximately 3-4 thousand” FORTRAN statements. The data set in Table 4.6 is from J. Musa’s (1979) “Software Reliability Data”, available from DACS, Rome Air Development Center, New York. The fourth set of data, DS4, which is shown in Table 4.7 contains 20 inter-events times which are Weibull distributed with scale parameter 10 and shape parameter 2 [Klefsjo and Kumar (1992)].
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Times</th>
<th>Cumulative Times</th>
<th>Error Number</th>
<th>Times</th>
<th>Cumulative Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>9</td>
<td>14</td>
<td>9</td>
<td>87</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>21</td>
<td>15</td>
<td>4</td>
<td>91</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>32</td>
<td>16</td>
<td>1</td>
<td>92</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>36</td>
<td>17</td>
<td>3</td>
<td>95</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>43</td>
<td>18</td>
<td>6</td>
<td>104</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>45</td>
<td>19</td>
<td>1</td>
<td>105</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>50</td>
<td>20</td>
<td>1</td>
<td>106</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>58</td>
<td>21</td>
<td>11</td>
<td>116</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>63</td>
<td>22</td>
<td>33</td>
<td>149</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>70</td>
<td>23</td>
<td>7</td>
<td>156</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>71</td>
<td>24</td>
<td>91</td>
<td>247</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>77</td>
<td>25</td>
<td>2</td>
<td>249</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>78</td>
<td>26</td>
<td>1</td>
<td>250</td>
</tr>
</tbody>
</table>

Table 4.4: NTDS data.

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Times</th>
<th>Cumulative Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
<td>1.1</td>
</tr>
<tr>
<td>3</td>
<td>0.65</td>
<td>1.75</td>
</tr>
<tr>
<td>4</td>
<td>1.90</td>
<td>3.65</td>
</tr>
<tr>
<td>5</td>
<td>1.59</td>
<td>5.24</td>
</tr>
<tr>
<td>6</td>
<td>8.34</td>
<td>14.07</td>
</tr>
<tr>
<td>7</td>
<td>9.94</td>
<td>24.01</td>
</tr>
<tr>
<td>8</td>
<td>7.25</td>
<td>31.60</td>
</tr>
<tr>
<td>9</td>
<td>8.34</td>
<td>39.90</td>
</tr>
<tr>
<td>10</td>
<td>3.86</td>
<td>43.46</td>
</tr>
<tr>
<td>11</td>
<td>13.11</td>
<td>56.57</td>
</tr>
<tr>
<td>12</td>
<td>34.15</td>
<td>90.72</td>
</tr>
<tr>
<td>13</td>
<td>82.7</td>
<td>173.42</td>
</tr>
<tr>
<td>14</td>
<td>1.1</td>
<td>174.52</td>
</tr>
<tr>
<td>15</td>
<td>51.59</td>
<td>226.11</td>
</tr>
</tbody>
</table>

Table 4.5: Data on F11-D program.
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Times</th>
<th>Cumulative Times</th>
<th>Error Number</th>
<th>Times</th>
<th>Cumulative Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>115</td>
<td>115</td>
<td>20</td>
<td>589</td>
<td>6150</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>118</td>
<td>21</td>
<td>15</td>
<td>6165</td>
</tr>
<tr>
<td>3</td>
<td>83</td>
<td>201</td>
<td>22</td>
<td>390</td>
<td>6555</td>
</tr>
<tr>
<td>4</td>
<td>178</td>
<td>379</td>
<td>23</td>
<td>1863</td>
<td>8418</td>
</tr>
<tr>
<td>5</td>
<td>194</td>
<td>573</td>
<td>24</td>
<td>1337</td>
<td>9755</td>
</tr>
<tr>
<td>6</td>
<td>136</td>
<td>709</td>
<td>25</td>
<td>4508</td>
<td>14263</td>
</tr>
<tr>
<td>7</td>
<td>1077</td>
<td>1786</td>
<td>26</td>
<td>834</td>
<td>15097</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>1801</td>
<td>27</td>
<td>3400</td>
<td>18497</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>1816</td>
<td>28</td>
<td>6</td>
<td>18503</td>
</tr>
<tr>
<td>10</td>
<td>92</td>
<td>1908</td>
<td>29</td>
<td>4561</td>
<td>23064</td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1958</td>
<td>30</td>
<td>3186</td>
<td>26250</td>
</tr>
<tr>
<td>12</td>
<td>71</td>
<td>2029</td>
<td>31</td>
<td>10571</td>
<td>36821</td>
</tr>
<tr>
<td>13</td>
<td>606</td>
<td>2635</td>
<td>32</td>
<td>563</td>
<td>37384</td>
</tr>
<tr>
<td>14</td>
<td>1189</td>
<td>3824</td>
<td>33</td>
<td>2770</td>
<td>40154</td>
</tr>
<tr>
<td>15</td>
<td>40</td>
<td>3864</td>
<td>34</td>
<td>652</td>
<td>40806</td>
</tr>
<tr>
<td>16</td>
<td>788</td>
<td>4652</td>
<td>35</td>
<td>5593</td>
<td>46399</td>
</tr>
<tr>
<td>17</td>
<td>222</td>
<td>4874</td>
<td>36</td>
<td>11696</td>
<td>58095</td>
</tr>
<tr>
<td>18</td>
<td>72</td>
<td>4946</td>
<td>37</td>
<td>6724</td>
<td>64819</td>
</tr>
<tr>
<td>19</td>
<td>615</td>
<td>5561</td>
<td>38</td>
<td>2546</td>
<td>67365</td>
</tr>
</tbody>
</table>

Table 4.6: DACS data.

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Times</th>
<th>Cumulative Times</th>
<th>Error Number</th>
<th>Times</th>
<th>Cumulative Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.8</td>
<td>3.8</td>
<td>11</td>
<td>19.0</td>
<td>96.9</td>
</tr>
<tr>
<td>2</td>
<td>9.3</td>
<td>13.1</td>
<td>12</td>
<td>5.5</td>
<td>102.3</td>
</tr>
<tr>
<td>3</td>
<td>4.5</td>
<td>17.6</td>
<td>13</td>
<td>3.2</td>
<td>105.5</td>
</tr>
<tr>
<td>4</td>
<td>0.8</td>
<td>18.4</td>
<td>14</td>
<td>4.0</td>
<td>109.5</td>
</tr>
<tr>
<td>5</td>
<td>8.6</td>
<td>27.0</td>
<td>15</td>
<td>9.1</td>
<td>118.6</td>
</tr>
<tr>
<td>6</td>
<td>11.0</td>
<td>38.0</td>
<td>16</td>
<td>16.5</td>
<td>135.1</td>
</tr>
<tr>
<td>7</td>
<td>7.1</td>
<td>45.1</td>
<td>17</td>
<td>7.7</td>
<td>142.8</td>
</tr>
<tr>
<td>8</td>
<td>8.1</td>
<td>53.2</td>
<td>18</td>
<td>10.9</td>
<td>153.7</td>
</tr>
<tr>
<td>9</td>
<td>11.4</td>
<td>64.6</td>
<td>19</td>
<td>7.3</td>
<td>161.0</td>
</tr>
<tr>
<td>10</td>
<td>13.3</td>
<td>77.9</td>
<td>20</td>
<td>15.5</td>
<td>176.5</td>
</tr>
</tbody>
</table>

Table 4.7: Times to the events (DS4).
4.5.2 Criteria for model's comparison

We use the following comparison criteria to compare several models' performance:

- Kolmogorov-Smirnov (KS) test statistic.
- Akaike information criterion (AIC).
- Chi-square ($\chi^2$) test statistic.
- The sum of square errors (SSE).

A brief description of each technique can be found in Chapter 3.

4.5.3 The models

In our application we study the following NHPP models:

1) K-stage Erlangian (gamma) growth curve model. Actually, the following four special cases of this model will be considered:

   a) K-model at K = 1 which give the exponential (G&O) SRGM,
   b) K-model at K = 2 which give the S-shaped SRGM,
   c) K-model at K = 3, and
   d) K-model at K = 4.

2) Modified exponential SRGM,

3) inflection S-shaped SRGM,

4) testing-effort dependent SRGM,

5) Littlewood SRGM,

6) Duane Model,

7) logistic SRGM, and

8) Gompertz SRGM. All these models were discussed, in details, in Section 4.2
### Estimation of parameters

Table 4.8 shows summary of data analysis. It contains the MLEs for all the underlying models. These estimates were evaluated by solving the likelihood equations in Table 4.2 for all these models, iteratively. The four data sets to be used for this purpose are in Tables 4.4, 4.5, 4.6 and 4.7.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data set one NTDS</th>
<th>Data set two F11-D program</th>
<th>Data set three DACS</th>
<th>Data set four DS4</th>
</tr>
</thead>
<tbody>
<tr>
<td>G &amp; O</td>
<td>(\hat{b}) 0.0058</td>
<td>(\hat{b}) 0.0149</td>
<td>38.6840 (5.99 \times 10^{-5})</td>
<td>66.8264 (0.0020)</td>
</tr>
<tr>
<td>S-shaped</td>
<td>(\hat{b}) 0.0186</td>
<td>(\hat{b}) 0.0334</td>
<td>38.0646 (1.28 \times 10^{-4})</td>
<td>24.3061 (0.0179)</td>
</tr>
<tr>
<td>K-model at (K = 3)</td>
<td>(\hat{b}) 0.0301</td>
<td>(\hat{b}) 0.0507</td>
<td>38.0082 (1.94 \times 10^{-4})</td>
<td>21.7398 (0.0320)</td>
</tr>
<tr>
<td>K-model at (K = 4)</td>
<td>(\hat{b}) 0.0411</td>
<td>(\hat{b}) 0.0677</td>
<td>38.0003 (2.58 \times 10^{-4})</td>
<td>20.8898 (0.0453)</td>
</tr>
<tr>
<td>Inflection</td>
<td>(\hat{b}) 0.0057</td>
<td>(\hat{b}) 0.0244</td>
<td>38.0000 (2.00 \times 10^{-4})</td>
<td>27.6823 (0.0150)</td>
</tr>
<tr>
<td>Duane</td>
<td>(\hat{b}) 0.2565</td>
<td>(\hat{b}) 1.7060</td>
<td>0.4540</td>
<td>0.1833</td>
</tr>
<tr>
<td>Logistic SRGM</td>
<td>(\hat{b}) 0.8365</td>
<td>(\hat{b}) 0.4010</td>
<td>0.3982</td>
<td>0.9071</td>
</tr>
<tr>
<td>Littlewood NHPP model</td>
<td>(\hat{b}) 0.1208</td>
<td>(\hat{b}) 0.3288</td>
<td>3.2162</td>
<td>11.2555</td>
</tr>
<tr>
<td>Modified exponential SRGM</td>
<td>(\hat{b}) 0.0260</td>
<td>(\hat{b}) 0.0250</td>
<td>9.67 (10^{-5})</td>
<td>0.0255</td>
</tr>
<tr>
<td>Testing-effort dependent SRGM</td>
<td>(\hat{b}) 1.3969</td>
<td>(\hat{b}) 1.5723</td>
<td>0.9404</td>
<td>1.4276</td>
</tr>
<tr>
<td>Gompertz SRGM</td>
<td>(\hat{b}) 26.6329</td>
<td>(\hat{b}) 21.0912</td>
<td>38.2406</td>
<td>24.4957</td>
</tr>
</tbody>
</table>

Table 4.8: Summary of data analyses.
4.5.5 Graphical description

Here we use all the underlying models to compare the prediction results using the above four data sets. The results are given in Tables 4.9 - 4.12. The plots of these results are shown in Figures 4.2.a - 4.2.d, respectively. From Figure 4.2.a, we can see by inspection that the S-shaped model appears to fit closest to the NTDS data points. Similarly, Figure 4.2.b shows that G & O fits closest to the data points for the F11-D program. We can see also, from Figures 4.2.c and 4.2.d, the Duane and Littlewood model fits the DACS and DS4 Data points respectively. Of course, these are subjective judgments and we will investigate placing this selection on a more scientific basis in the next section.

The estimated expected number of remaining faults, computed from (4.12) for estimated values of the parameters in Table 4.8 for all the models, is shown in Figures 4.3.a - 4.3.d. From these Figures we see that the expected number of remaining faults decreases monotonically with time. Figure 4.3.a shows that the estimated expected number of remaining faults from the inflection model are not fitted at all to the actual number of remaining faults. Figure 4.3.d shows a similar effect for the G & O and Littlewood models. In particular the variation in these predictions of remaining faults indicates it is premature to make a reliability prediction from the NTDS and DS4 data sets.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1</td>
<td>1.7261</td>
<td>0.3441</td>
<td>0.0717</td>
<td>0.0152</td>
<td>0.6999</td>
<td>1.6116</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>3.8920</td>
<td>1.6202</td>
<td>0.6988</td>
<td>0.3060</td>
<td>1.6662</td>
<td>3.2741</td>
</tr>
<tr>
<td>32</td>
<td>3</td>
<td>5.7494</td>
<td>3.3019</td>
<td>1.9499</td>
<td>1.1664</td>
<td>2.5850</td>
<td>4.6571</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>6.3961</td>
<td>3.9877</td>
<td>2.5495</td>
<td>1.6497</td>
<td>2.9270</td>
<td>5.1394</td>
</tr>
<tr>
<td>43</td>
<td>5</td>
<td>7.4922</td>
<td>5.2458</td>
<td>3.7485</td>
<td>2.7064</td>
<td>3.5353</td>
<td>5.9630</td>
</tr>
<tr>
<td>45</td>
<td>6</td>
<td>7.7977</td>
<td>5.6145</td>
<td>4.1204</td>
<td>3.0538</td>
<td>3.7115</td>
<td>6.1941</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>8.5449</td>
<td>6.5466</td>
<td>5.0951</td>
<td>3.9995</td>
<td>4.1564</td>
<td>6.7648</td>
</tr>
<tr>
<td>58</td>
<td>8</td>
<td>9.6968</td>
<td>8.0484</td>
<td>6.7511</td>
<td>5.6997</td>
<td>4.8814</td>
<td>7.6591</td>
</tr>
<tr>
<td>63</td>
<td>9</td>
<td>0.3901</td>
<td>8.9810</td>
<td>7.8209</td>
<td>6.8457</td>
<td>5.3428</td>
<td>8.2077</td>
</tr>
<tr>
<td>70</td>
<td>10</td>
<td>1.3277</td>
<td>10.2645</td>
<td>9.3311</td>
<td>8.5102</td>
<td>5.9991</td>
<td>8.9639</td>
</tr>
<tr>
<td>116</td>
<td>21</td>
<td>16.6276</td>
<td>17.4399</td>
<td>17.9603</td>
<td>18.3568</td>
<td>10.6017</td>
<td>13.6773</td>
</tr>
</tbody>
</table>

Table 4.9: The actual data and reliability prediction results for NTDS data.
<table>
<thead>
<tr>
<th>Times</th>
<th>Error No.</th>
<th>Logistic</th>
<th>Littlewood</th>
<th>Modified Exponential</th>
<th>Testing-effort dependent</th>
<th>Gompertz</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1</td>
<td>2.5291</td>
<td>1.7338</td>
<td>5.0557</td>
<td>21.6283</td>
<td>0.7629</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>3.4290</td>
<td>3.9032</td>
<td>10.0352</td>
<td>22.7441</td>
<td>1.6681</td>
</tr>
<tr>
<td>32</td>
<td>3</td>
<td>4.4800</td>
<td>5.7585</td>
<td>13.3406</td>
<td>23.3005</td>
<td>2.9345</td>
</tr>
<tr>
<td>43</td>
<td>5</td>
<td>5.7715</td>
<td>7.4956</td>
<td>15.8053</td>
<td>23.6909</td>
<td>4.6007</td>
</tr>
<tr>
<td>45</td>
<td>6</td>
<td>6.0331</td>
<td>7.7995</td>
<td>16.1834</td>
<td>23.7510</td>
<td>4.9407</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>6.7228</td>
<td>8.5434</td>
<td>17.0507</td>
<td>23.8901</td>
<td>5.8320</td>
</tr>
<tr>
<td>58</td>
<td>8</td>
<td>7.9286</td>
<td>9.6887</td>
<td>18.2386</td>
<td>24.0860</td>
<td>7.3554</td>
</tr>
<tr>
<td>63</td>
<td>9</td>
<td>8.7413</td>
<td>10.3776</td>
<td>18.8768</td>
<td>24.1951</td>
<td>8.3488</td>
</tr>
<tr>
<td>77</td>
<td>12</td>
<td>11.2069</td>
<td>12.2024</td>
<td>20.3347</td>
<td>24.4596</td>
<td>11.1819</td>
</tr>
<tr>
<td>78</td>
<td>13</td>
<td>11.3909</td>
<td>12.3271</td>
<td>20.4236</td>
<td>24.4766</td>
<td>11.3828</td>
</tr>
<tr>
<td>116</td>
<td>21</td>
<td>18.2267</td>
<td>16.5691</td>
<td>22.8441</td>
<td>24.9983</td>
<td>18.0917</td>
</tr>
<tr>
<td>156</td>
<td>23</td>
<td>23.0198</td>
<td>20.1448</td>
<td>24.2786</td>
<td>25.3861</td>
<td>22.4969</td>
</tr>
<tr>
<td>247</td>
<td>24</td>
<td>25.9780</td>
<td>25.8554</td>
<td>25.9646</td>
<td>25.9843</td>
<td>25.9599</td>
</tr>
<tr>
<td>249</td>
<td>25</td>
<td>25.9929</td>
<td>25.9520</td>
<td>25.9883</td>
<td>25.9948</td>
<td>25.9869</td>
</tr>
<tr>
<td>250</td>
<td>26</td>
<td>26.0000</td>
<td>26.0000</td>
<td>26.0000</td>
<td>26.0000</td>
<td>26.0000</td>
</tr>
</tbody>
</table>

Table 4.9: Continued.
Table 4.10: The actual data and reliability prediction results for data on F11-D program.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>1</td>
<td>0.1153</td>
<td>0.0021</td>
<td>0.00004</td>
<td>0.0001</td>
<td>0.0374</td>
</tr>
<tr>
<td>1.1</td>
<td>2</td>
<td>0.2525</td>
<td>0.0099</td>
<td>0.0004</td>
<td>0.0012</td>
<td>0.0827</td>
</tr>
<tr>
<td>1.75</td>
<td>3</td>
<td>0.3997</td>
<td>0.0247</td>
<td>0.0016</td>
<td>0.0019</td>
<td>0.1321</td>
</tr>
<tr>
<td>3.65</td>
<td>4</td>
<td>0.8221</td>
<td>0.1030</td>
<td>0.0138</td>
<td>0.0041</td>
<td>0.2794</td>
</tr>
<tr>
<td>5.24</td>
<td>5</td>
<td>1.1664</td>
<td>0.2049</td>
<td>0.0038</td>
<td>0.0075</td>
<td>0.4056</td>
</tr>
<tr>
<td>14.07</td>
<td>6</td>
<td>2.9373</td>
<td>1.2207</td>
<td>0.5361</td>
<td>0.2431</td>
<td>1.1564</td>
</tr>
<tr>
<td>24.01</td>
<td>7</td>
<td>4.6711</td>
<td>2.8848</td>
<td>1.8624</td>
<td>1.2350</td>
<td>2.0985</td>
</tr>
<tr>
<td>31.26</td>
<td>8</td>
<td>5.7832</td>
<td>4.2168</td>
<td>3.1885</td>
<td>2.4660</td>
<td>2.8437</td>
</tr>
<tr>
<td>39.60</td>
<td>9</td>
<td>6.9224</td>
<td>5.7325</td>
<td>4.8782</td>
<td>4.2245</td>
<td>3.7509</td>
</tr>
<tr>
<td>43.46</td>
<td>10</td>
<td>7.4037</td>
<td>6.4048</td>
<td>5.6700</td>
<td>5.0964</td>
<td>4.1850</td>
</tr>
<tr>
<td>56.57</td>
<td>11</td>
<td>8.8462</td>
<td>8.4742</td>
<td>8.1973</td>
<td>7.9879</td>
<td>5.6981</td>
</tr>
<tr>
<td>226.11</td>
<td>15</td>
<td>15.0000</td>
<td>15.0000</td>
<td>15.0000</td>
<td>15.0000</td>
<td>15.0000</td>
</tr>
</tbody>
</table>

Table 4.10: Continued.

<table>
<thead>
<tr>
<th>Times</th>
<th>Error No.</th>
<th>Logistic</th>
<th>Littlewood</th>
<th>Modified Exponential</th>
<th>Testing-effort dependent</th>
<th>Gompertz</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>1</td>
<td>3.5716</td>
<td>0.1216</td>
<td>0.2631</td>
<td>10.6209</td>
<td>2.2603</td>
</tr>
<tr>
<td>1.1</td>
<td>2</td>
<td>3.6128</td>
<td>0.2663</td>
<td>0.5710</td>
<td>11.1773</td>
<td>2.3165</td>
</tr>
<tr>
<td>1.75</td>
<td>3</td>
<td>3.6577</td>
<td>0.4214</td>
<td>0.8952</td>
<td>11.5089</td>
<td>2.3781</td>
</tr>
<tr>
<td>3.65</td>
<td>4</td>
<td>3.7914</td>
<td>0.8654</td>
<td>1.7896</td>
<td>12.0382</td>
<td>2.5624</td>
</tr>
<tr>
<td>5.24</td>
<td>5</td>
<td>3.9057</td>
<td>1.2266</td>
<td>2.4814</td>
<td>12.3000</td>
<td>2.7213</td>
</tr>
<tr>
<td>24.01</td>
<td>7</td>
<td>5.4126</td>
<td>4.8545</td>
<td>7.8568</td>
<td>13.4044</td>
<td>4.8303</td>
</tr>
<tr>
<td>31.26</td>
<td>8</td>
<td>6.0595</td>
<td>5.9868</td>
<td>9.0412</td>
<td>13.5951</td>
<td>5.7046</td>
</tr>
<tr>
<td>43.46</td>
<td>10</td>
<td>7.1971</td>
<td>7.6192</td>
<td>10.4350</td>
<td>13.8326</td>
<td>7.1608</td>
</tr>
<tr>
<td>90.72</td>
<td>12</td>
<td>11.3270</td>
<td>11.6604</td>
<td>12.8734</td>
<td>14.3587</td>
<td>11.5995</td>
</tr>
<tr>
<td>226.11</td>
<td>15</td>
<td>15.0000</td>
<td>15.0000</td>
<td>15.0000</td>
<td>15.0000</td>
<td>15.0000</td>
</tr>
<tr>
<td>Times</td>
<td>Error No.</td>
<td>G &amp; O model</td>
<td>S-shaped Model</td>
<td>K-model at K = 3</td>
<td>K-model at K = 4</td>
<td>Inflection model</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>-------------</td>
<td>----------------</td>
<td>------------------</td>
<td>------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>115</td>
<td>1</td>
<td>0.2656</td>
<td>0.0041</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.1760</td>
</tr>
<tr>
<td>118</td>
<td>2</td>
<td>0.2725</td>
<td>0.0043</td>
<td>0.0003</td>
<td>0.0002</td>
<td>0.1806</td>
</tr>
<tr>
<td>201</td>
<td>3</td>
<td>0.4630</td>
<td>0.0121</td>
<td>0.0004</td>
<td>0.0003</td>
<td>0.3092</td>
</tr>
<tr>
<td>379</td>
<td>4</td>
<td>0.8683</td>
<td>0.0436</td>
<td>0.0024</td>
<td>0.0004</td>
<td>0.5891</td>
</tr>
<tr>
<td>573</td>
<td>5</td>
<td>1.3052</td>
<td>0.0980</td>
<td>0.0080</td>
<td>0.0007</td>
<td>0.9008</td>
</tr>
<tr>
<td>709</td>
<td>6</td>
<td>1.6085</td>
<td>0.1483</td>
<td>0.0148</td>
<td>0.0015</td>
<td>1.1234</td>
</tr>
<tr>
<td>1786</td>
<td>7</td>
<td>3.9248</td>
<td>0.8594</td>
<td>0.0206</td>
<td>0.0049</td>
<td>3.0043</td>
</tr>
<tr>
<td>1801</td>
<td>8</td>
<td>3.9560</td>
<td>0.8728</td>
<td>0.0207</td>
<td>0.0051</td>
<td>3.0320</td>
</tr>
<tr>
<td>1816</td>
<td>9</td>
<td>3.9872</td>
<td>0.8863</td>
<td>0.0212</td>
<td>0.0052</td>
<td>3.0597</td>
</tr>
<tr>
<td>1908</td>
<td>10</td>
<td>4.1779</td>
<td>0.9709</td>
<td>0.2427</td>
<td>0.0632</td>
<td>3.2304</td>
</tr>
<tr>
<td>1958</td>
<td>11</td>
<td>4.2811</td>
<td>1.0181</td>
<td>0.2605</td>
<td>0.0694</td>
<td>3.3237</td>
</tr>
<tr>
<td>2029</td>
<td>12</td>
<td>4.4271</td>
<td>1.0868</td>
<td>0.2869</td>
<td>0.0789</td>
<td>3.4571</td>
</tr>
<tr>
<td>2635</td>
<td>13</td>
<td>5.6483</td>
<td>1.7427</td>
<td>0.5767</td>
<td>0.1985</td>
<td>4.6298</td>
</tr>
<tr>
<td>3824</td>
<td>14</td>
<td>7.9193</td>
<td>3.3727</td>
<td>1.4915</td>
<td>0.6937</td>
<td>7.0972</td>
</tr>
<tr>
<td>3864</td>
<td>15</td>
<td>7.9929</td>
<td>3.3861</td>
<td>1.5302</td>
<td>0.7174</td>
<td>7.1836</td>
</tr>
<tr>
<td>4652</td>
<td>16</td>
<td>9.4079</td>
<td>4.6026</td>
<td>2.3936</td>
<td>1.2892</td>
<td>8.9264</td>
</tr>
<tr>
<td>4874</td>
<td>17</td>
<td>9.7947</td>
<td>4.9623</td>
<td>2.6698</td>
<td>1.4869</td>
<td>9.4295</td>
</tr>
<tr>
<td>4946</td>
<td>18</td>
<td>9.9190</td>
<td>5.0803</td>
<td>2.7624</td>
<td>1.5545</td>
<td>9.5937</td>
</tr>
<tr>
<td>8418</td>
<td>23</td>
<td>15.3202</td>
<td>11.1816</td>
<td>8.5289</td>
<td>6.6767</td>
<td>17.7516</td>
</tr>
<tr>
<td>23064</td>
<td>29</td>
<td>28.9670</td>
<td>30.2529</td>
<td>31.2626</td>
<td>32.1125</td>
<td>36.1854</td>
</tr>
<tr>
<td>26250</td>
<td>30</td>
<td>30.6551</td>
<td>32.3386</td>
<td>33.5281</td>
<td>34.4330</td>
<td>37.0229</td>
</tr>
<tr>
<td>36821</td>
<td>31</td>
<td>34.4216</td>
<td>36.1321</td>
<td>36.9866</td>
<td>37.4414</td>
<td>37.8800</td>
</tr>
<tr>
<td>37384</td>
<td>32</td>
<td>34.5629</td>
<td>36.2441</td>
<td>37.0677</td>
<td>37.4972</td>
<td>37.8928</td>
</tr>
<tr>
<td>40154</td>
<td>33</td>
<td>35.1930</td>
<td>36.7105</td>
<td>37.3852</td>
<td>37.7027</td>
<td>37.9385</td>
</tr>
<tr>
<td>40806</td>
<td>34</td>
<td>35.3267</td>
<td>36.8022</td>
<td>37.4434</td>
<td>37.7378</td>
<td>37.9460</td>
</tr>
<tr>
<td>46399</td>
<td>35</td>
<td>36.2824</td>
<td>37.3779</td>
<td>37.7682</td>
<td>37.9131</td>
<td>37.9825</td>
</tr>
<tr>
<td>58095</td>
<td>36</td>
<td>37.4922</td>
<td>37.8785</td>
<td>37.9708</td>
<td>37.9931</td>
<td>37.9986</td>
</tr>
<tr>
<td>64819</td>
<td>37</td>
<td>37.8873</td>
<td>37.9781</td>
<td>37.9957</td>
<td>37.9992</td>
<td>37.9998</td>
</tr>
<tr>
<td>67365</td>
<td>38</td>
<td>38.0000</td>
<td>38.0000</td>
<td>38.0000</td>
<td>38.0000</td>
<td>38.0000</td>
</tr>
</tbody>
</table>

Table 4.11: The actual data and reliability prediction results for DACS data.
<table>
<thead>
<tr>
<th>Times</th>
<th>Error No.</th>
<th>Logistic</th>
<th>Littlewood</th>
<th>Modified Exponentia</th>
<th>Testing-effort dependent</th>
<th>Gompertz</th>
</tr>
</thead>
<tbody>
<tr>
<td>115</td>
<td>1</td>
<td>9.1329</td>
<td>0.2788</td>
<td>1.5521</td>
<td>26.9522</td>
<td>5.7075</td>
</tr>
<tr>
<td>118</td>
<td>2</td>
<td>9.1350</td>
<td>0.2860</td>
<td>1.5913</td>
<td>26.9965</td>
<td>5.7103</td>
</tr>
<tr>
<td>201</td>
<td>3</td>
<td>9.1908</td>
<td>0.4859</td>
<td>2.6512</td>
<td>27.9160</td>
<td>5.7870</td>
</tr>
<tr>
<td>379</td>
<td>4</td>
<td>9.3114</td>
<td>0.9109</td>
<td>4.7699</td>
<td>29.0209</td>
<td>5.9533</td>
</tr>
<tr>
<td>573</td>
<td>5</td>
<td>9.4441</td>
<td>1.3685</td>
<td>6.8592</td>
<td>29.7452</td>
<td>6.1369</td>
</tr>
<tr>
<td>1786</td>
<td>7</td>
<td>10.3019</td>
<td>4.1030</td>
<td>16.0025</td>
<td>31.7463</td>
<td>7.3398</td>
</tr>
<tr>
<td>1801</td>
<td>8</td>
<td>10.3128</td>
<td>4.1355</td>
<td>16.0532</td>
<td>31.7511</td>
<td>7.3553</td>
</tr>
<tr>
<td>1816</td>
<td>9</td>
<td>10.3238</td>
<td>4.1679</td>
<td>16.1633</td>
<td>31.7757</td>
<td>7.3707</td>
</tr>
<tr>
<td>1908</td>
<td>10</td>
<td>10.3909</td>
<td>4.3663</td>
<td>16.6412</td>
<td>31.8627</td>
<td>7.4657</td>
</tr>
<tr>
<td>2029</td>
<td>12</td>
<td>10.4796</td>
<td>4.6254</td>
<td>17.2365</td>
<td>31.9710</td>
<td>7.5914</td>
</tr>
<tr>
<td>2635</td>
<td>13</td>
<td>10.9308</td>
<td>5.8930</td>
<td>19.7927</td>
<td>32.4311</td>
<td>8.2324</td>
</tr>
<tr>
<td>3824</td>
<td>14</td>
<td>11.8489</td>
<td>8.2404</td>
<td>22.9360</td>
<td>33.0856</td>
<td>9.5402</td>
</tr>
<tr>
<td>3864</td>
<td>15</td>
<td>11.8805</td>
<td>8.3163</td>
<td>23.0165</td>
<td>33.1039</td>
<td>9.5852</td>
</tr>
<tr>
<td>4652</td>
<td>16</td>
<td>12.5127</td>
<td>9.7719</td>
<td>24.3535</td>
<td>33.4294</td>
<td>10.4829</td>
</tr>
<tr>
<td>4874</td>
<td>17</td>
<td>12.6939</td>
<td>10.1689</td>
<td>24.6597</td>
<td>33.5110</td>
<td>10.7392</td>
</tr>
<tr>
<td>4946</td>
<td>18</td>
<td>12.7529</td>
<td>10.2964</td>
<td>24.7536</td>
<td>33.5367</td>
<td>10.8227</td>
</tr>
<tr>
<td>5561</td>
<td>19</td>
<td>13.2627</td>
<td>11.3619</td>
<td>25.4633</td>
<td>33.7418</td>
<td>11.5403</td>
</tr>
<tr>
<td>6150</td>
<td>20</td>
<td>13.7597</td>
<td>12.3437</td>
<td>26.0195</td>
<td>33.9177</td>
<td>12.2346</td>
</tr>
<tr>
<td>6165</td>
<td>21</td>
<td>13.7724</td>
<td>12.3682</td>
<td>26.0325</td>
<td>33.9220</td>
<td>12.2524</td>
</tr>
<tr>
<td>6555</td>
<td>22</td>
<td>14.1060</td>
<td>12.9976</td>
<td>26.3497</td>
<td>34.0291</td>
<td>12.7150</td>
</tr>
<tr>
<td>8418</td>
<td>23</td>
<td>15.7418</td>
<td>15.7964</td>
<td>27.5249</td>
<td>34.4649</td>
<td>14.9369</td>
</tr>
<tr>
<td>9755</td>
<td>24</td>
<td>16.9494</td>
<td>17.6095</td>
<td>28.1720</td>
<td>34.7209</td>
<td>16.5219</td>
</tr>
<tr>
<td>15097</td>
<td>26</td>
<td>21.8498</td>
<td>23.4965</td>
<td>30.1801</td>
<td>35.4755</td>
<td>22.4352</td>
</tr>
<tr>
<td>18497</td>
<td>27</td>
<td>24.8240</td>
<td>26.3335</td>
<td>31.2296</td>
<td>35.8243</td>
<td>25.6441</td>
</tr>
<tr>
<td>18503</td>
<td>28</td>
<td>24.8290</td>
<td>26.3379</td>
<td>31.2313</td>
<td>35.8248</td>
<td>25.6493</td>
</tr>
<tr>
<td>23064</td>
<td>29</td>
<td>28.3656</td>
<td>29.3101</td>
<td>32.4521</td>
<td>36.2013</td>
<td>29.1578</td>
</tr>
<tr>
<td>26250</td>
<td>30</td>
<td>30.4399</td>
<td>30.9408</td>
<td>33.1958</td>
<td>36.4214</td>
<td>31.0928</td>
</tr>
<tr>
<td>36821</td>
<td>31</td>
<td>34.9794</td>
<td>34.5507</td>
<td>35.1447</td>
<td>36.9935</td>
<td>35.1471</td>
</tr>
<tr>
<td>37384</td>
<td>32</td>
<td>35.1355</td>
<td>34.6856</td>
<td>35.2296</td>
<td>37.0190</td>
<td>35.2857</td>
</tr>
<tr>
<td>40154</td>
<td>33</td>
<td>35.8065</td>
<td>35.2872</td>
<td>35.6237</td>
<td>37.1390</td>
<td>35.8853</td>
</tr>
<tr>
<td>40806</td>
<td>34</td>
<td>35.9431</td>
<td>35.4150</td>
<td>35.7111</td>
<td>37.1661</td>
<td>36.0083</td>
</tr>
<tr>
<td>46399</td>
<td>35</td>
<td>36.8452</td>
<td>36.3292</td>
<td>36.3837</td>
<td>37.3811</td>
<td>36.8363</td>
</tr>
<tr>
<td>58095</td>
<td>36</td>
<td>37.7397</td>
<td>37.4977</td>
<td>37.4282</td>
<td>37.7552</td>
<td>37.7141</td>
</tr>
<tr>
<td>64819</td>
<td>37</td>
<td>37.9496</td>
<td>37.8873</td>
<td>37.8610</td>
<td>37.9364</td>
<td>37.9422</td>
</tr>
<tr>
<td>67365</td>
<td>38</td>
<td>38.0000</td>
<td>38.0000</td>
<td>38.0000</td>
<td>38.0000</td>
<td>38.0000</td>
</tr>
</tbody>
</table>

Table 4.11: Continued
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8</td>
<td>1</td>
<td>0.5097</td>
<td>0.0536</td>
<td>0.0059</td>
<td>0.0007</td>
<td>0.3201</td>
<td>0.6152</td>
</tr>
<tr>
<td>13.1</td>
<td>2</td>
<td>1.7409</td>
<td>0.5710</td>
<td>0.1949</td>
<td>0.0674</td>
<td>1.1488</td>
<td>1.8903</td>
</tr>
<tr>
<td>17.6</td>
<td>3</td>
<td>2.3284</td>
<td>0.9780</td>
<td>0.4256</td>
<td>0.1873</td>
<td>1.5728</td>
<td>2.4709</td>
</tr>
<tr>
<td>18.4</td>
<td>4</td>
<td>2.4323</td>
<td>1.0591</td>
<td>0.4773</td>
<td>0.2175</td>
<td>1.6497</td>
<td>2.5726</td>
</tr>
<tr>
<td>27.0</td>
<td>5</td>
<td>3.5386</td>
<td>2.0656</td>
<td>1.2369</td>
<td>0.7472</td>
<td>2.5058</td>
<td>3.6428</td>
</tr>
<tr>
<td>38.0</td>
<td>6</td>
<td>4.9260</td>
<td>3.6126</td>
<td>2.6888</td>
<td>2.0122</td>
<td>3.6755</td>
<td>4.9667</td>
</tr>
<tr>
<td>45.1</td>
<td>7</td>
<td>5.8052</td>
<td>4.7016</td>
<td>3.8395</td>
<td>3.1458</td>
<td>4.4719</td>
<td>5.8015</td>
</tr>
<tr>
<td>53.2</td>
<td>8</td>
<td>6.7931</td>
<td>5.9845</td>
<td>5.2800</td>
<td>4.6622</td>
<td>5.4162</td>
<td>6.7392</td>
</tr>
<tr>
<td>64.6</td>
<td>9</td>
<td>8.1564</td>
<td>7.8013</td>
<td>7.4096</td>
<td>7.0200</td>
<td>6.8005</td>
<td>8.0369</td>
</tr>
</tbody>
</table>

Table 4.12: The actual data and reliability prediction results for DS4 data.
<table>
<thead>
<tr>
<th>Times</th>
<th>Error No.</th>
<th>Logistic</th>
<th>Littlewood</th>
<th>Modified Exponential</th>
<th>Testing-effort dependent</th>
<th>Gompertz</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8</td>
<td>1</td>
<td>2.0068</td>
<td>0.5151</td>
<td>2.5499</td>
<td>16.0913</td>
<td>1.1764</td>
</tr>
<tr>
<td>13.1</td>
<td>2</td>
<td>2.4841</td>
<td>1.7574</td>
<td>7.2534</td>
<td>17.3485</td>
<td>1.7756</td>
</tr>
<tr>
<td>17.6</td>
<td>3</td>
<td>2.7489</td>
<td>2.3494</td>
<td>8.9388</td>
<td>17.6505</td>
<td>2.1230</td>
</tr>
<tr>
<td>18.4</td>
<td>4</td>
<td>2.7985</td>
<td>2.4540</td>
<td>9.2067</td>
<td>17.6959</td>
<td>2.1887</td>
</tr>
<tr>
<td>27.0</td>
<td>5</td>
<td>3.3810</td>
<td>3.5672</td>
<td>11.6017</td>
<td>18.0885</td>
<td>2.9678</td>
</tr>
<tr>
<td>38.0</td>
<td>6</td>
<td>4.2672</td>
<td>4.9608</td>
<td>13.7121</td>
<td>18.4383</td>
<td>4.1455</td>
</tr>
<tr>
<td>45.1</td>
<td>7</td>
<td>4.9280</td>
<td>5.8428</td>
<td>14.6985</td>
<td>18.6135</td>
<td>4.9981</td>
</tr>
<tr>
<td>53.2</td>
<td>8</td>
<td>5.7679</td>
<td>6.8326</td>
<td>15.5816</td>
<td>18.7824</td>
<td>6.0408</td>
</tr>
<tr>
<td>64.6</td>
<td>9</td>
<td>7.0986</td>
<td>8.1968</td>
<td>16.5237</td>
<td>18.9806</td>
<td>7.5955</td>
</tr>
<tr>
<td>77.9</td>
<td>10</td>
<td>8.8374</td>
<td>9.7470</td>
<td>17.3353</td>
<td>19.1714</td>
<td>9.4674</td>
</tr>
<tr>
<td>96.9</td>
<td>11</td>
<td>11.5229</td>
<td>11.8876</td>
<td>18.1751</td>
<td>19.3934</td>
<td>12.0932</td>
</tr>
<tr>
<td>102.3</td>
<td>12</td>
<td>12.2942</td>
<td>12.4806</td>
<td>18.3682</td>
<td>19.4485</td>
<td>12.8060</td>
</tr>
<tr>
<td>105.5</td>
<td>13</td>
<td>12.7470</td>
<td>12.8289</td>
<td>18.4753</td>
<td>19.4797</td>
<td>13.2188</td>
</tr>
<tr>
<td>142.8</td>
<td>17</td>
<td>17.3669</td>
<td>16.7242</td>
<td>19.4348</td>
<td>19.7863</td>
<td>17.3687</td>
</tr>
<tr>
<td>153.7</td>
<td>18</td>
<td>18.3854</td>
<td>17.8080</td>
<td>19.6417</td>
<td>19.8606</td>
<td>18.3322</td>
</tr>
<tr>
<td>161.0</td>
<td>19</td>
<td>18.9755</td>
<td>18.5206</td>
<td>19.7665</td>
<td>19.9074</td>
<td>18.9156</td>
</tr>
<tr>
<td>176.5</td>
<td>20</td>
<td>20.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td>20.0000</td>
</tr>
</tbody>
</table>

Table 4.12: Continued
Figure 4.2: Data analysis based on some SRGMs.

Figure 4.2.a: NTDS data. S-shaped model chosen.

Figure 4.2.b: Data on F11-D program. G&O model chosen.
Figure 4.2.c: DACS data. Duane model chosen.

Figure 4.2.d: DS4 data. Littlewood model chosen.
Figure 4.3: Estimated expected number of remaining faults based on some SRGMs

Figure 4.3.a: NTDS data.

Figure 4.3.b: F11-D program.
Figure 4.3.c: DACS data.

Figure 4.3.d: DS4 data.
4.5.6 Choice between software reliability models

In this Section we analyze the four data sets given in Tables 4.4, 4.5, 4.6 and 4.7 to measure how well a set of data fits a particular model using AIC, K-S, $\chi^2$ and SSE techniques. For this purpose we will use the eleven models that we investigate. The parameters of these models have been estimated as mentioned previously.

For the four data sets we can see in Table 4.13, the best model that fits a certain data set is obtained by choosing the lowest AIC value. Although we have mentioned in Section 4.1 that there are good reasons for favoring this as a fitness measure, there is no absolute reference for judging which is the ‘best’ fitness measure. Khoshgoftaar and Woodcock (1992) illustrated its use. Although they made some general claims about the results they obtained, four of the example data sets they used were from similar students projects, using students of similar engineering background, following similar development and test methods. We feel this limits the generality of their conclusions, and prefer to use four independent test data sets in our experiments.

In Table 4.14 the values of the test statistics in equation (3.3) are compared with the critical values $D_{n-1,\alpha}$ with sample size $n-1$. The value of $\alpha$ is taken to be 0.05. If the calculated value $D$ is less than the selected critical value, then it can be concluded that the observed data fits the applied SRGM. It can be see that:

1) G&O, S-shaped, logistic, Littlewood, Gompertz, $K = 3$ and $K = 4$ models fit the NTDS data at $\alpha = 0.05$. Also the Duane model fits the same data at $\alpha = 0.01$.
2) G&O, S-shaped, Duane, logistic, Littlewood, modified exponential and Gompertz fit the data on F11-D program at $\alpha = 0.05$, Also $K = 3$ and $K = 4$ models fit the same data at $\alpha = 0.01$.
3) Only Duane model fits the DACS data at $\alpha = 0.05$ but also G&O, inflection logistic, Littlewood, modified exponential and Gompertz do at $\alpha = 0.01$.
4) All the models fit the DS4 data at 0.05, but the modified exponential and testing-effort dependent model did not fit this data set at all.
Table 4.15 shows the values of $\chi^2$ statistic. If we compare the calculated values of $\chi^2$ for different models corresponding to each data set with the critical values in $\chi^2$-Table at $\alpha = 0.05$, Duane was the only model that fitted all the data sets.

Table 4.16 presents the values of sum of square errors fitness measures. The lowest SSE value indicates the best model for certain data, according to this technique the testing-effort dependent model performed the worst for all the four selected data sets.
<table>
<thead>
<tr>
<th>Model</th>
<th>NTDS Data</th>
<th>Rank</th>
<th>Data on F11-D Program</th>
<th>Rank</th>
<th>DACS Data</th>
<th>Rank</th>
<th>DS4 Data</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>G&amp;O</td>
<td>185.367</td>
<td>8th</td>
<td>105.3723</td>
<td>2nd</td>
<td>612.970</td>
<td>3rd</td>
<td>224.5471</td>
<td>8th</td>
</tr>
<tr>
<td>S-shaped</td>
<td>168.290</td>
<td>1st</td>
<td>128.0588</td>
<td>6th</td>
<td>662.473</td>
<td>6th</td>
<td>141.3904</td>
<td>5th</td>
</tr>
<tr>
<td>K = 3</td>
<td>169.203</td>
<td>2nd</td>
<td>156.1142</td>
<td>7th</td>
<td>725.132</td>
<td>7th</td>
<td>142.2292</td>
<td>6th</td>
</tr>
<tr>
<td>K = 4</td>
<td>173.611</td>
<td>6th</td>
<td>186.1309</td>
<td>9th</td>
<td>792.772</td>
<td>8th</td>
<td>152.9158</td>
<td>7th</td>
</tr>
<tr>
<td>Inflection</td>
<td>1449.20</td>
<td>11th</td>
<td>782.6545</td>
<td>11th</td>
<td>4328.97</td>
<td>11th</td>
<td>1067.5463</td>
<td>11th</td>
</tr>
<tr>
<td>Duane</td>
<td>172.815</td>
<td>5th</td>
<td>97.9918</td>
<td>1st</td>
<td>603.629</td>
<td>1st</td>
<td>130.9069</td>
<td>1st</td>
</tr>
<tr>
<td>Logistic</td>
<td>176.909</td>
<td>7th</td>
<td>124.5272</td>
<td>5th</td>
<td>660.226</td>
<td>5th</td>
<td>140.8258</td>
<td>4th</td>
</tr>
<tr>
<td>Littlewood</td>
<td>171.428</td>
<td>4th</td>
<td>105.9637</td>
<td>3rd</td>
<td>612.788</td>
<td>2nd</td>
<td>132.8863</td>
<td>2nd</td>
</tr>
<tr>
<td>Modified</td>
<td>592.030</td>
<td>10th</td>
<td>330.4724</td>
<td>10th</td>
<td>1962.33</td>
<td>10th</td>
<td>481.6838</td>
<td>10th</td>
</tr>
<tr>
<td>Testing-effort</td>
<td>350.451</td>
<td>9th</td>
<td>180.2318</td>
<td>8th</td>
<td>814.939</td>
<td>9th</td>
<td>270.5670</td>
<td>9th</td>
</tr>
<tr>
<td>Gompertz</td>
<td>169.829</td>
<td>3rd</td>
<td>118.3928</td>
<td>4th</td>
<td>137.650</td>
<td>4th</td>
<td>137.6501</td>
<td>3rd</td>
</tr>
</tbody>
</table>

Table 4.13: Summary of AIC values for some SRGMs.
Table 4.4: Summary of D values for some SKMs at α = 0.05 and α = 0.01.

<table>
<thead>
<tr>
<th>Model</th>
<th>Lognormal</th>
<th>Exponential</th>
<th>Weibull</th>
<th>Data</th>
<th>Status</th>
<th>DACS</th>
<th>Data</th>
<th>Status</th>
<th>Program</th>
<th>Test</th>
<th>Status</th>
<th>DACS</th>
<th>Data</th>
<th>Status</th>
<th>NTDs</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>0.219</td>
<td>*</td>
<td></td>
<td>0.315</td>
<td>**</td>
<td>0.716</td>
<td>**</td>
<td>0.266</td>
<td>**</td>
<td>0.019</td>
<td></td>
<td>0.364</td>
<td>**</td>
<td>0.978</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>**</td>
<td>0.019</td>
<td>*</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>**</td>
<td>0.019</td>
<td>**</td>
<td>0.019</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>0.364</td>
<td></td>
</tr>
<tr>
<td>**</td>
<td>0.019</td>
<td>*</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>**</td>
<td>0.019</td>
<td>**</td>
<td>0.019</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>0.364</td>
<td></td>
</tr>
<tr>
<td>**</td>
<td>0.019</td>
<td>*</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>**</td>
<td>0.019</td>
<td>**</td>
<td>0.019</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>0.364</td>
<td></td>
</tr>
<tr>
<td>**</td>
<td>0.019</td>
<td>*</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>**</td>
<td>0.019</td>
<td>**</td>
<td>0.019</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>0.364</td>
<td></td>
</tr>
<tr>
<td>**</td>
<td>0.019</td>
<td>*</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>**</td>
<td>0.019</td>
<td>**</td>
<td>0.019</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>0.364</td>
<td></td>
</tr>
<tr>
<td>**</td>
<td>0.019</td>
<td>*</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>**</td>
<td>0.019</td>
<td>**</td>
<td>0.019</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>0.364</td>
<td></td>
</tr>
<tr>
<td>**</td>
<td>0.019</td>
<td>*</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>**</td>
<td>0.019</td>
<td>**</td>
<td>0.019</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>0.364</td>
<td></td>
</tr>
<tr>
<td>**</td>
<td>0.019</td>
<td>*</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>**</td>
<td>0.019</td>
<td>**</td>
<td>0.019</td>
<td></td>
<td>0.019</td>
<td>**</td>
<td>0.364</td>
<td>0.364</td>
<td></td>
</tr>
</tbody>
</table>
Table 4.15: Summary of $\chi^2$ values for some SCGMS at $\alpha = 0.05$:

The model does not fit the data set.
The model fits the data set at $\alpha = 0.05$.

<table>
<thead>
<tr>
<th>Model</th>
<th>Complete</th>
<th>Test Stat.</th>
<th>Exponential</th>
<th>Logistic</th>
<th>Infection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0623</td>
<td>6.0695</td>
<td>9.7623</td>
<td>4.4483</td>
<td>1.7968</td>
<td></td>
</tr>
<tr>
<td>1.040185</td>
<td>3.3267</td>
<td>1.1765</td>
<td>4.2653</td>
<td>6.2569</td>
<td></td>
</tr>
<tr>
<td>0.436096</td>
<td>0.519055</td>
<td>0.164935</td>
<td>0.0400</td>
<td>1.4738</td>
<td></td>
</tr>
<tr>
<td>3.1402</td>
<td>6.2599</td>
<td>0.548094</td>
<td>0.15294</td>
<td>0.2944</td>
<td></td>
</tr>
<tr>
<td>5.1532</td>
<td>2.7524</td>
<td>2.9322</td>
<td>3.4526</td>
<td>2.5738</td>
<td></td>
</tr>
<tr>
<td>9.4764</td>
<td>2.0572</td>
<td>1.7868</td>
<td>4.2259</td>
<td>2.1588</td>
<td></td>
</tr>
<tr>
<td>9.756</td>
<td>1.7032</td>
<td>1.4707</td>
<td>3.4212</td>
<td>1.3542</td>
<td></td>
</tr>
<tr>
<td>9.483</td>
<td>1.2367</td>
<td>1.1394</td>
<td>3.2401</td>
<td>1.0968</td>
<td></td>
</tr>
<tr>
<td>3.2401</td>
<td>0.2832</td>
<td>0.3536</td>
<td>3.4532</td>
<td>0.5343</td>
<td></td>
</tr>
<tr>
<td>3.2401</td>
<td>0.2832</td>
<td>0.3536</td>
<td>3.4532</td>
<td>0.5343</td>
<td></td>
</tr>
<tr>
<td>3.2401</td>
<td>0.2832</td>
<td>0.3536</td>
<td>3.4532</td>
<td>0.5343</td>
<td></td>
</tr>
<tr>
<td>3.2401</td>
<td>0.2832</td>
<td>0.3536</td>
<td>3.4532</td>
<td>0.5343</td>
<td></td>
</tr>
</tbody>
</table>

Note: The table includes various statistical models and their corresponding $\chi^2$ values, indicating the fit of the models to the data set at a significance level of 0.05.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Rank</th>
<th>Rank</th>
<th>Rank</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
</tbody>
</table>

**Table 4.16: Summary of SSE values for some SGRMs**

<table>
<thead>
<tr>
<th>Program</th>
<th>Rank</th>
<th>Data</th>
<th>Rank</th>
<th>Rank</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>DACS Data</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>NIDS Data</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>Model</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
</tbody>
</table>

**Composites**

- Testing-error dependent
- Modified Exponential
- Logistic
- Dunne
- Infection
- S-shaped
- G&O
4.5.7 Conclusion

In current software development methodologies, faults in programs are generally assumed to be randomly distributed through the software system. A failure is considered corrected when the fault causing it has been removed. When a program is in the testing phase of development, the number of remaining faults and the time until the occurrence of the next fault can be forecast by a software reliability model. There are several different ways to model a software failure process. One class of these models is that of the non-homogeneous Poisson model. Software reliability growth models fit failure data, providing predictions of reliability using the assumption that as the time between failures increases, so the number of residual faults decreases.

In our experiments, we consider eleven SRGMs. The models studied here fit the data on time to failure. Time to failure is assumed to be:

- a random variable.
- an indicator of the number of residual faults.

The validity of these assumptions is dependant on the nature of the test process being used during the reliability testing phase. The best model is the one that predicts the future stochastic behaviour of the system failure process better than the others.

Inferences, by using the maximum likelihood method, are used for estimating the model parameters and consequently their characteristics. Attempts to compare different models to find the best model among existing ones for each application on a set of data is performed, numerically and graphically, by using (i) goodness-of-fit techniques, (ii) reliability prediction.

The graphical representations show that the test data is inherently noisy. That is, there are some unknown variables that are influencing whether or not a fault is revealed in the failure data. Hence, it may not be possible to find a single model that is guaranteed to give a reliability prediction with a high level of confidence. Perhaps we should consider providing a distribution of possible values for the reliability distribution. Also these plots indicate that
different judgments about the quality of the software could be made, depending on the choice of the model. So no one model provides a best fit in all cases. This is not entirely unexpected as the time between failures is a function of both

- number and distribution of faults, and
- the nature of the testing process.

Both of these vary to an extent from project to project. So, we need an objective model selection method. The methods used in this study are AIC, K-S, $\chi^2$ and SSE. Unfortunately, the ranking varies according to the test used. We may not be able to select a preferred model, but all the models provide some useful information about the software under test.

From Table 4.8 we can show that:

1) For NTDS data $\hat{\alpha}$ varies between 0.2565 and 67.3215.
2) For Data on F11-D Program $\hat{\alpha}$ varies between 1.7060 and 21.0912.
3) For DACS Data $\hat{\alpha}$ varies between 0.4540 and 53.6386.
4) For DS4 data $\hat{\alpha}$ varies between 0.1833 and 92.3388.

With the infinite failure Duane model we can not make any relation between $a$, $b$ and the physical characteristics of the system. The testing-effort dependent model is consistently scored as a bad performer across all data sets: with AIC it is ranked 9th on three data sets and 8th on one, with SSE it was ranked 11th on all data sets. With D values, it was ranked out on all data sets. This model makes a very specific assumption about the distribution of testing-effort during the test phase, as we have discussed. These results indicate that this assumption is not valid for the test data that is used here. Inflection is also a consistently poor performer. It assumes a mutual dependency between software faults. Again these results indicate this assumption is not valid in these cases. According to these fitness measures and the analysis of assumptions we could exclude those three models. If we do this, we get a range of predictions of $\hat{\alpha}$ varying between 26.2233 and 36.0146 for NTDS data, between 15.0025 and 15.9529 for Data on F11-D program, between 38.0011 and 38.7546 for DACS Data and between 20.8898 and 92.3388 for DS4 data. From these results we could conclude that:
- A stable prediction can be made from the data on the F11-D program and the DACS data. So, it is appropriate to use the data to make a release decision.

- With NTDS and DS4 data there is still a wide variation, so we need to gain more information before making a release decision.

- Also we can see that Duane is the best fit to DS4 data where there is a wide divergence of the prediction of residual defects, which also indicates it is premature to make a reliability prediction for this data set. So the Duane model can provide a useful safety check when making reliability predictions — since it is an infinite failure model, the conclusions to draw when this model fits the data are obvious!

According to the previous selection method, although with some differences in the preferences, perhaps we need:

- to make the uncertainties associated with reliability estimation more explicit.
- to factor into the prediction process, more information about the effectiveness of the processes being used.
Chapter 5

On a General Formulation of the Littlewood Model

5.1 Introduction

In the last chapter, we explored the parameter estimation and performance of a number of SRGMs. We will now undertake a theoretical analysis in order to identify some important generalizations of existing models. This will then enable us to produce a general purpose software tool that can be configured to provide estimators for a range of different SRGMs. Finally, we evaluate the predictive accuracy of these models.

During the last thirty-five years, many software reliability models and measurement procedures have been proposed for prediction, estimation and engineering of software reliability [Goel (1985), Xie (1991)]. These models can be used not only to estimate the current reliability but also to predict their future values. The paper which is most cited in the software reliability modeling is by Jelinski and Moranda (1972). Their model (JM) has been widely discussed theoretically and practically since its publication. Times between failures are assumed to be exponential with a parameter that is proportional to the number of remaining faults. This model has later been both modified and generalized by various authors.

In (1981) Littlewood modified the JM model by suggesting that not all faults have the same severity. To reflect his idea he assumed that whenever a failure occurs, the fault causing this failure is discovered, and directly corrected. Failures due to different faults occur
independently of each other. The time till failure due to the i-th fault is assumed to be a random variable, having exponential distribution with a random failure rate, $\Lambda_i$ (i=1, 2, …..,N). All the N failure rates are independent random variables, having a common gamma distribution. As a result of this modification, he found that the times till failures of the N faults are independent random variables having a common Pareto distribution of the second kind. In this Chapter we adopt the same idea, but in our case to modify the general form of the JM model; the Weibull model. As a result of our modification we obtain a Burr type XII model, which is the general form of the Littlewood model.

Today the number of existing models exceeds two-hundred with more and more models being developed every year. However, still there is no generally accepted standard, and there is limited confirmation of any model presently being used on an ongoing basis. Models that are good in general are not always the best choice for a particular data set, and it is not possible to know in advance what model should be used in any existing application. One of the good strategies when we aim to choose one appropriate model to a particular application is to study several models. This increases the possibility of finding one suitable model, but in the same time consumes much time and effort. Therefore, when we are to study reliability models of the type we discuss in this Chapter it is worthwhile to study Burr type XII and Weibull models and get several models as special cases, so avoiding duplication of effort and time.

In this Chapter we conduct a comparative study between six special cases of Burr type XII and Weibull models based on two failure data sets. For the purpose of model estimation we use the maximum likelihood method. Different selecting techniques [model accuracy (PLR), model bias (u-plot), and bias trend (y-plot)] are used to assess the effectiveness of the selected models. Our study reveals the potential problem of obtaining poor raw prediction results, and not being able to trust any of the models. As a solution, the use of new approaches that may help in enhancing these poor prediction results is suggested.
5.2 Theoretical and mathematical work that support the generalization

The Jelinski and Moranda (JM) model for software reliability growth is one of the most commonly cited. This model assumes that the failure rate of a program is a constant multiple of the number of residual faults. This means that all faults impact equally the failure rate of the software. In (1981) Littlewood proposed a modified version of this model by assuming that not all faults have the same seriousness. His model assumes that times between failures are exponentially distributed with a parameter that is treated as a random variable, having a Gamma prior distribution. In this section we show that if we modify the Weibull model by following the same procedure as Littlewood, we will obtain a Burr Type XII model which is the general case of Littlewood L model. Actually we illustrate the generalization in three steps:

- Firstly, we follow exactly Littlewood's work using the basic assumption of independent execution time between failures with exponential distribution, and independent failure rate having a gamma distribution. This, of course, leads to independent times till failures having a common Pareto distribution of the second type.
- Secondly, we change the assumption of the time between failures to become independent and having a Rayleigh distribution and as a result of that we obtained independent times till failures having a special case of a Burr Type XII distribution.
- Finally, the distribution of the independent times between failures is changed to become a Weibull distribution and this leads to the general form of the Littlewood L model. The details are in the following sub-sections.

5.2.1 Jelinski and Moranda (JM) model

The most commonly used model for analyzing software failure data was originally introduced by Jalinski and Moranda (JM) (1972). No account is taken of the internal structure of the program. The only input to the model is the sequence of execution times between successive failures: \(X_1, X_2, \ldots\). The objective was to estimate current and future reliability on the basis of these past interfailure times. The problem, then, is one of estimating and predicting reliability growth. The assumptions made in the JM model are:
• Assumption 1: The random variables \( X_i = S_{(i)} - S_{(i-1)} \), \( i = 1, 2, \ldots, N \), representing successive interfailure execution times, are independent, with exponential distributions:

\[
f(x_i|\lambda_i) = \lambda_i \exp(-\lambda_i x_i) \quad ; \quad \lambda_i > 0 \quad , \quad x_i > 0
\]

• Assumption 2: At each failure, a fault is fixed immediately, with the result that the failure rate improves. All such improvements are of equal size so that

\[
\lambda_i = (N - i + 1) \phi
\]

since at this state (i-1) faults have been eliminated. \( N \) is the initial number of faults in the program, \( \phi \) is the improvement in failure rate at each fix and \( 0 = S_{(0)} \leq S_{(1)} \leq \ldots \leq S_{(N)} \) are the ordered failure times.

Spreij (1985), Joe and Reid (1985) observed that the likelihood under this model is the same as the likelihood under the assumption that \( X_1, X_2, \ldots, X_N \) are independently exponentially distributed with a common parameter \( \phi \).

In the early papers describing variants of the model, it is suggested that the maximum likelihood method be used to estimate \( N \) and \( \phi \) from the inter-event time data. These estimates can then be substituted into appropriate expressions to make reliability predictions.

5.2.1.1 Littlewood modification to the JM model

Littlewood (1981) modified the JM model suggesting that the time till failure due to the \( i^{th} \) fault is assumed to be a random variable, having an exponential distribution with a random failure rate, \( \Lambda_i \) (\( i = 1, 2, \ldots, N \)). His assumptions basically were:

i) Each of the \( N \) faults in the program will cause a failure after a time which is distributed exponentially, and independently of other faults, with rate \( \phi \).
ii) When a failure occurs, there is an instantaneous removal of the fault which caused the failure.

iii) If a total time $t$ has elapsed, and $i$ faults have been removed, the failure rate of the program is

$$\Lambda = \Phi_1 + \Phi_2 + \ldots + \Phi_{N-i},$$

(5.1)

where $\Phi_1, \Phi_2, \ldots, \Phi_{N-i}$ are i.i.d.

iv) When debugging starts (i.e., $t = 0$) each $\Phi$ has the pdf $\beta\text{GAM}(\beta\phi; \alpha)$, as we will prove later, where $\text{GAM}(x; \alpha)$ is a gamma pdf, $x^{\alpha-1}e^{-x}/\Gamma(\alpha)$.

Assume that the random variables $S_1, S_2, \ldots$ are independent. If the failure rate at any time were known, it might be reasonable to assume [Jelinski and Moranda (1972), Shooman (1973) and Musa (1975)] that failures occurred randomly with that failure rate.

Consider the random variable $S$, when total elapsed execution time is $t$ and $i$ faults have been fixed. There are $(N-i)$ faults remaining. The crucial point is that these $(N-i)$ faults will have different occurrence rates, $\phi_1, \phi_2, \ldots, \phi_{N-i}$. The failure rate of the program is now

$$\lambda = \phi_1 + \phi_2 + \ldots + \phi_{N-i}.$$

Figure 5.1
Assuming exponentially distributed times to failure for each fault, we could describe the current reliability of the program completely if the $\phi$'s were known. The $\phi$'s however, are not known. In fact, there will not even be any failure data available to estimate the $\phi$'s for remaining faults. We must therefore model our uncertainty about a $\phi$ value by treating each occurrence rate as a random variable $\Phi$ with some distribution. The failure rate of the program is then given by assumption (iii).

Now consider the occurrence rate, $\Phi$, of one of the $N-i$ remaining faults at the epoch "now" in Figure 5.1.

\[
\text{pdf}\{\phi \mid \text{this fault not fixed in } (0, \tau)\} = c \text{ } \exp(-\phi \tau) \cdot \beta^\alpha \phi^{\alpha-1} \cdot \exp(-\phi \beta)/\Gamma(\alpha)
\]

\[
= (\beta + \tau)^\alpha \phi^{\alpha-1} \exp\{- (\beta + \tau) \phi \}/\Gamma(\alpha)
\]

\[
= (\beta + \tau) \text{GAM}\{\alpha; (\beta + \tau)\phi\}
\]

where $c$ is constant and $\pi(\phi)$ is the prior density of $\Phi$.

From equation (5.1), the failure rate of the program, $\Lambda$, is a sum of $(N - i)$ i.i.d. $\text{GAM}\{\alpha; (\beta + \tau)\phi\}$ random variables, and so has pdf $(\beta + \tau) \text{GAM}\{(N-i)\alpha; (\beta + \tau)\lambda\}$. At $\tau = 0$ we get assumption (iv).

Now we can obtain the pdf of the time to next failure $X$

\[
f(x) = \frac{(\beta + \tau)^{(N-i)\alpha}}{\Gamma[(N-i)\alpha]} \int_0^\infty \lambda \exp(-\lambda x) \lambda^{(N-i)\alpha-1} \exp\{-\lambda(\beta + \tau)\} \text{ d}\lambda
\]

\[
= \frac{\alpha(N-i)/(\beta + \tau)}{\left[1 + \frac{x}{\beta + \tau}\right]^{(N-i)\alpha+1}}
\]

(5.2)
which is $(\beta + \tau)^{-1} \text{Par}\left(\frac{x}{\beta + \tau}, (N-i)\alpha\right)$,

where $\text{Par}(x; \alpha)$ is Pareto pdf, $\alpha/(1+x)^{\alpha+1}$, $(x \geq 0)$. The cdf is given by:

$$F(x) = 1 - \left(\frac{x}{\beta + \tau + x}\right)^{(N-i)\alpha},$$

(5.3)

and the failure rate is

$$\lambda(x) = \frac{(N-i)\alpha}{(\beta + \tau + x)}.$$  

(5.4)

According to this model, the times till failure of the $N$ faults are independent random variables $X_1, X_2, \ldots, X_N$, with $N$ unknown, having a common Pareto distribution of the second kind, with a probability density function

$$f(x, \alpha, \beta) = \frac{\alpha}{\beta} \left(1 + \frac{x}{\beta}\right)^{-\alpha-1}, \quad 0 < x < \infty.$$  

(5.5)

It should be mentioned here that, both the JM and Littlewood models share the assumptions of:

1) perfect debugging,
2) independence of $X_1, X_2, \ldots, X_N$ and
3) the marginal distributions of $X_1, X_2, \ldots, X_N$ are identical.

### 5.2.2 Rayleigh model

Assume that the successive interfailure execution times are independent random variables, having a Rayleigh distribution.

$$f(x) = 2\lambda x \exp[-\lambda x^2]; \quad x > 0, \ (\lambda, \eta > 0).$$  

(5.6)
Each detected fault is immediately corrected, so the number of errors decreases by one. And the failure rate has the following formula:

\[ \dot{f}(x) = 2\lambda x. \quad (5.7) \]

Along with equation 5.6 the reliability and the mean time to failure become:

\[ R(x) = \exp[-\lambda x^2]. \quad (5.8) \]
\[ E(x) = \frac{1}{2} \sqrt{\pi/\lambda}. \quad (5.9) \]

While the median time to failure MTTF for this model can be evaluated by:

\[ \text{MTTF}(x) = \left(\frac{\lambda^{-1}}{\ln 2}\right)^{\frac{1}{2}}. \quad (5.10) \]

### 5.2.2.1 Modification to the Rayleigh model

Following the same procedure as in Section 5.2.1, we assume that the time till failure due to the \(i\)th fault has a Rayleigh distribution and consider the same assumption about \(\Lambda\). Then the density function of the \(i\)th interfailure time will be

\[ f(x|\tau;\Lambda = \lambda) = 2\lambda (x+\tau) \exp[-\lambda \left\{ (x+\tau)^2 - \tau^2 \right\}], \quad x > 0. \quad (5.11) \]

\[ \therefore \text{pdf}\{\Phi|\text{this fault is not fixed in}(0,\tau)\} \]
\[ = c \text{P}\{\text{no failure caused by this fault in}(0,\tau)\mid\Phi = \phi\} \cdot \pi(\phi) \]
\[ = c \beta^\alpha \phi^{\alpha-1} \exp(-\phi \tau^2) \cdot \exp(-\beta \phi)/\Gamma(\alpha). \]
\[ = (\beta + \tau^2)^{\alpha} \phi^{\alpha-1} \exp\left\{ -\phi \left( \beta + \tau^2 \right) \right\}/\Gamma(\alpha). \quad (5.12) \]
which is a GAM \( \{a; (\beta + \tau^2)\} \). Then, the failure rate of the program, \( \Lambda \), is a sum of \((N-i)\) i.i.d. GAM \( \{a; (\beta + \tau^2)\} \) random variables and so it has a pdf

\[
h_1(\lambda) = \frac{(\beta + \tau^2)^{(N-i)\alpha}}{\Gamma [(N-i)\alpha]} \lambda^{(N-i)\alpha - 1} \exp\{ -\lambda (\beta + \tau^2) \}. \tag{5.13}
\]

Consequently, the pdf of \( X \) is given by:

\[
f_1(x) = \int_0^\infty 2 \lambda x \exp(-\lambda x^2) \lambda^{(N-i)\alpha - 1} \exp\{ -\lambda (\beta + \tau^2) \} \, d\lambda.
\]

\[
f_1(x) = \frac{2x(N-i)\alpha/(\beta + \tau^2)}{\left[1 + x^2/(\beta + \tau^2)\right]^{(N-i)\alpha + 1}}, \quad x > 0, \tag{5.14}
\]

which is a density function of a special case of Burr Type XII distribution. The corresponding cdf is in the form:

\[
F_1(x) = 1 - \left(1 + \frac{x^2}{\beta + \tau^2}\right)^{-(N-i)\alpha}, \tag{5.15}
\]

and the failure rate is

\[
\lambda_1(x) = \frac{2(N-i)\alpha x}{(\beta + \tau^2 + x^2)}. \tag{5.16}
\]

According to this model, the times till failure of the \( N \) faults are independent random variables \( X_1, X_2, \ldots, X_N \) (units on test), having a special Burr distribution with a pdf

\[
f(x; \alpha_1, \beta_1) = \frac{2 \alpha_1 x/\beta_1}{\left(1 + x^2/\beta_1\right)^{\alpha_1 + 1}}, \quad x > 0. \tag{5.17}
\]

And the cdf is
\[ F(x; \alpha_1, \beta_1) = 1 - \left(1 + \frac{x^2}{\beta_1^n}\right)^{-\alpha_1} ; \quad x > 0 \quad , \] (5.18)

with mean given by

\[ E(X) = \alpha_1 \sqrt{\frac{3}{2}} \beta_1 B\left(\alpha_1, \frac{1}{2}\right) \quad , \] (5.19)

where \( B(\alpha, \beta) \) is the ordinary beta function defined by

\[ \int_0^1 y^{\alpha-1} (1 - y)^{\beta-1} dy . \]

### 5.2.3 Weibull model

Now assume that the successive interfailure execution times are independent random variables with a Weibull distribution. Then the density function of the \( i^{th} \) interfailure time will be

\[ f(x; a, b) = ab x^{b-1} \exp\left(-ax^b\right) ; \quad x > 0; (a, b > 0) . \] (5.20)

This model assumes a failure rate function of the form:

\[ fr(x) = ab x^{b-1} . \] (5.21)

The reliability, which is defined as the probability that the software experienced no failures until time \( t \) is given by:

\[ R(x) = \exp[-ax^b] . \] (5.22)

And the expected amount of time that the software operates before failure is also easily accomplished using the following formula:

\[ E(x) = b^{-1} a^{-1/n} \Gamma(b^{-1}) . \] (5.23)
5.2.3.1 Modification to the Weibull model

Following Littlewood's (1981) assumption about \( \Lambda \), then the density function of the \( i^{th} \) interfailure time will be

\[
g(x|\tau, \Lambda, \lambda) = \lambda b (x+\tau) \exp\{-\lambda(x+\tau)^b - \tau^b\}, \quad x > 0. \tag{5.24}
\]

\[
\therefore \text{pdf}(\phi | \text{this fault not fixed in } (0, \tau)) = c \Phi(0) \pi(\phi)
\]

\[
= c \Phi(X > \tau) \pi(\phi)
\]

\[
= c \exp(-\phi \tau^b) \phi^{a-1} \exp(-\beta \phi)/\Gamma(\alpha)
\]

\[
= \left(\tau^b + \beta\right)^{\alpha} \phi^{a-1} \exp\{-\left(\tau^b + \beta\right)\phi\}/\Gamma(\alpha), \quad \phi > 0; \quad (b, \alpha, \beta > 0), \tag{5.25}
\]

which is also \( \text{GAM}\{\alpha; \left(\tau^b + \beta\right)\phi\} \). The pdf of \( \Lambda \) is

\[
h_2(\lambda) = \frac{(\beta + \tau^b)^{(N-i)\alpha}}{\Gamma\{(N-i)\alpha\}} \lambda^{(N-i)\alpha-1} \exp\{-\lambda(\beta + \tau^b)\}, \quad , \tag{5.26}
\]

and the pdf of \( X \) is given by

\[
f_2(x) = \int_0^\infty a b x^{b-1} \exp(-ax^b) \lambda^{(N-i)\alpha-1} \exp\{-\lambda(\beta + \tau^b)\} \, d\lambda.
\]

\[
= \frac{x^{b-1} (N-i)\alpha/(\beta + \tau^b)}{(1+x^b/(\beta + \tau^b))^{(N-i)\alpha+1}, \quad x > 0. \tag{5.27}
\]

which is a density function of a Burr type XII distribution. The cdf is:
\[ F_2(x) = 1 - \left( 1 + \frac{x^b}{(\beta + x^b)} \right)^{-\alpha} \] , \hspace{1cm} (5.28)

and the failure rate is

\[ \lambda_2(x) = \frac{(N-i)\alpha x^{b-1}}{\beta + x^b} \] . \hspace{1cm} (5.29)

According to this model, the times till failure of the N faults are independent random variables \( X_1, X_2, \ldots, X_N \) (units on test) having a common three parameter Burr distribution with pdf

\[ f(x; \alpha_2, \beta_2, \gamma_2) = \frac{\alpha_2 \gamma_2 x^{\gamma_2 - 1}}{\beta_2 \left( 1 + \frac{x^{\gamma_2}}{\beta_2} \right)^{\alpha_2 + 1}} \] , \hspace{1cm} (5.30)

and cdf

\[ F(x) = 1 - \left( 1 + \frac{x^{\gamma_2}}{\beta_2} \right)^{-\alpha_2} \] . \hspace{1cm} (5.31)

The pdf and some important reliability measures of the Weibull and Burr type XII models are presented in Table 5.1.
<table>
<thead>
<tr>
<th>Measure of reliability</th>
<th>Probability density function $f(x_1)$</th>
<th>Reliability $R(x_1)$</th>
<th>Failure rate $fr(x_1)$</th>
<th>Mean time to next failure $\mu_{ITF(x_1)}$</th>
<th>Median time to next failure $MTTF(x_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Weibull model</strong></td>
<td>ab $x^{b-1} \exp(-ax^b)$; $x &gt; 0$, $(a, b &gt; 0)$</td>
<td>$\exp(-ax^b)$</td>
<td>$ab x^{b-1}$</td>
<td>$b^{-1}a^{-1/b}\Gamma(b^{-1})$</td>
<td>$(1/\ln 2)^{1/b}$</td>
</tr>
<tr>
<td><strong>Burr type XII model</strong></td>
<td>$\alpha_2 \gamma_2 x^{\gamma_2-1} / \beta_2$ $(1 + x^{\gamma_2} / \beta_2)^{-\alpha_2}$; $x &gt; 0$, $(\alpha_2, \beta_2, \gamma_2 &gt; 0)$</td>
<td>$(1 + x^{\gamma_2} / \beta_2)^{-\alpha_2}$</td>
<td>$\frac{\alpha_2 \gamma_2 x^{\gamma_2-1}}{\beta_2 + x^{\gamma_2}}$</td>
<td>$\beta_2^{1/\gamma_2} \Gamma(\alpha_2 - \gamma_2) \Gamma(\gamma_2^{-1}) / \gamma_2 \Gamma(\alpha_2)$</td>
<td>$[\beta_2(2^{1/\alpha_2} - 1)]^{\gamma_2}$</td>
</tr>
</tbody>
</table>

Table 5.1: Summary of some characteristics of Weibull and Burr type XII models.
5.3 Software failure data and analysis

In this section we will evaluate the predictive validity of six conventional SRGMs to two failure data sets.

5.3.1 Techniques for the analysis predictive accuracy

Even though many software reliability models are available in the literature, no clear guidelines have been presented for choosing a particular model. A set of five criteria, for comparing software reliability models, were proposed in [Musa and Okomoto (1983)]. The most important of these criteria is predictive validity, and it is the criterion we consider here. Classification of a model requires applying it to many data sets and assessing its predictive validity. The predictive validity of a model can be estimated using various techniques. In our application we will use the following techniques to analyze the predictive validity of the underlying models:

a) The u-plot.
b) The y-plot.
c) The Prequential Likelihood Ratio (PLR).

A brief description of each technique can be found in Chapter 3.

5.3.2 The models

In our application, we study six conventional software reliability growth models. Using the theoretical results of Section 5.2, three of them (JM, R, and W) are studied as special cases of the Weibull model. The pdf of these three models are obtained by giving the parameter \( b \) in equation (5.20) the values (1, 2 and 3) respectively. Following Section 5.2.3, the rest of the models (L, SB and THPB) are studied as special cases of Burr Type XII model. By replacing the parameter \( \gamma_2 \) in equation (5.30) with the values (1, 2 and 3) correspondingly, we can obtain the pdf of these three models.
The main advantages of studying these models as special cases of the Weibull and Burr Type XII models, respectively, is to avoid duplication of effort and time. The pdf and some important reliability measures of all the studied special cases of the Weibull and Burr type XII models are presented in Tables 5.2 and 5.3, respectively.

<table>
<thead>
<tr>
<th>Probability density function $f(x_i)$</th>
<th>Reliability $R(x_i)$</th>
<th>Failure rate $fr(x_i)$</th>
<th>Mean time to next failure $\mu_{TTF}(x_i)$</th>
<th>Median time to next failure $MTTF(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>JM model $a \exp(-ax)$</td>
<td>$\exp(-ax)$</td>
<td>$a$</td>
<td>$1/a$</td>
<td>$1/a \ln 2$</td>
</tr>
<tr>
<td>R model $2a x \exp(-a x^2)$</td>
<td>$\exp(-ax^2)$</td>
<td>$2a x$</td>
<td>$\frac{1}{2} \left(\frac{\pi}{a}\right)^{\frac{1}{2}}$</td>
<td>$(1/a \ln 2)^{1/2}$</td>
</tr>
<tr>
<td>W model $3a x^2 \exp(-a x^3)$</td>
<td>$\exp(-ax^3)$</td>
<td>$3a x^2$</td>
<td>$\frac{1}{3} a^{-1/3} \Gamma\left(\frac{1}{3}\right)$</td>
<td>$(1/a \ln 2)^{1/3}$</td>
</tr>
</tbody>
</table>

Table 5.2: Weibull model in cases of $b = 1, 2, \text{ and } 3$ respectively.
<table>
<thead>
<tr>
<th>Probability density function $f(x_1)$</th>
<th>Reliability $R(x_1)$</th>
<th>Failure rate $fr(x_1)$</th>
<th>Mean time to next failure $\mu TTF(x_1)$</th>
<th>Median time to next failure $MTTF(x_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>L model $\frac{\alpha_2 / \beta_2}{(1 + x / \beta_2)^{\alpha_2 + 1}}$</td>
<td>$(1 + x / \beta_2)^{-\alpha_2}$</td>
<td>$\frac{\alpha_2}{\beta_2 + x}$</td>
<td>$\frac{\beta_2}{\alpha_2 - 1}$</td>
<td>$\beta_2 \left(2^{1/\alpha_2} - 1\right)$</td>
</tr>
<tr>
<td>SB model $\frac{2 \alpha_2 x / \beta_2}{(1 + x^2 / \beta_2)^{\alpha_2 + 1}}$</td>
<td>$(1 + x^2 / \beta_2)^{-\alpha_2}$</td>
<td>$\frac{2\alpha_2 x}{\beta_2 + x^2}$</td>
<td>$\sqrt{\pi \beta_2 \Gamma(\frac{\alpha_2 - 1}{2}) \Gamma(\alpha_2)} \frac{2}{2\Gamma(\alpha_2)}$</td>
<td>$\left[\beta_2 \left(2^{1/\alpha_2} - 1\right)\right]^{1/2}$</td>
</tr>
<tr>
<td>THPB model $\frac{3 \alpha_2 x^2 / \beta_2}{(1 + x^3 / \beta_2)^{\alpha_2 + 1}}$</td>
<td>$(1 + x^3 / \beta_2)^{-\alpha_2}$</td>
<td>$\frac{3\alpha_2 x^2}{\beta_2 + x^3}$</td>
<td>$\frac{1/3\beta_2^{1/3} \Gamma(\alpha_2 - 1/3) \Gamma(1/3)}{\Gamma(\alpha_2)}$</td>
<td>$\left[\beta_2 \left(2^{1/\alpha_2} - 1\right)\right]^{1/3}$</td>
</tr>
</tbody>
</table>

Table 5.3: Burr type XII model in cases of $\gamma_2 = 1, 2, \text{ and } 3$ respectively.

### 5.3.3 Data sets

In our application, the performance of the selected SRGMs will be examined using DACS and DS4 data sets (more details about these two data sets can be found in Chapter 4).

### 5.3.4 Parameter estimation

Parameter estimation is of primary importance in software reliability prediction. In our study we apply the maximum likelihood estimation MLE method, the most important and widely used estimation technique, to get the estimate of the underlying models. This technique has several properties including consistency, efficiency and asymptotic normality. We now derive the estimators for our generalized models.
5.3.4.1 Estimation the parameters of the Weibull model

The likelihood function for estimating the unknown two parameters \( N, \phi \) in the Weibull model is

\[
L(N, \phi) = \prod_{i=1}^{n} (N - i + 1)^{\phi} x_i^{\eta - 1} \exp[-(N - i + 1)\eta]\phi].
\] (5.32)

The MLE of \( N (\hat{N}) \) is obtained by numerically solving the following equation

\[
n \sum_{i=1}^{n} x_i^\eta = \sum_{i=1}^{n} (N - i + 1)x_i^\eta \left( \sum_{i=1}^{n} \frac{1}{N - i + 1} \right).
\] (5.33)

By substituting \( \hat{N} \) in the following equation we can obtain the MLE of \( \phi (\hat{\phi}) \).

\[
\phi = \frac{n}{\sum_{i=1}^{n} (N - i + 1)x_i^\eta}.
\] (5.34)

5.3.4.2 Estimation the parameters of the Burr type XII model

In this case the likelihood equations for the unknown parameters based upon the \( n \) observed times intervals \( t_1, t_2, \ldots, t_n, n \leq N \) is given by

\[
L(N, \alpha, \beta) = \prod_{i=1}^{n} f(x_i) = \eta^n \alpha^n \prod_{i=1}^{n} \left( \frac{N - i}{\beta + \eta x_i^{\eta - 1}} \right)^{(N-i)\alpha - 1} (1 + \frac{x_i^\eta}{\beta + \eta x_i^{\eta - 1}})^{-(N-i)\alpha - 1}.
\] (5.35)

The parameters to be estimated are \( N, \alpha \) and \( \beta \). The estimates of these three parameters can be obtained as follows. First, solving the following two equations in order to obtain the MLE of the parameters \( N, \beta \)
\[ \sum_{i=1}^{n} \frac{1}{N-i} = n \sum_{i=1}^{n} \log \left( \frac{1 + \frac{x_i^\eta}{\beta + \tau_{i-1}^\eta}}{\sum_{i=1}^{n} (N-i) \log \left( \frac{1 + \frac{x_i^\eta}{\beta + \tau_{i-1}^\eta}}{1 + \frac{x_i^\eta}{\beta + \tau_{i-1}^\eta}} \right) \right) \] \quad (5.36)

\[ n \sum_{i=1}^{n} \left( \beta + \tau_{i-1}^\eta \right) = \frac{n \sum_{i=1}^{n} (i-N)x_i^\eta \left( \beta + \tau_{i-1}^\eta \right)^2 \left( 1 + \frac{x_i^\eta}{\beta + \tau_{i-1}^\eta} \right)^2 - \sum_{i=1}^{n} x_i^\eta}{\sum_{i=1}^{n} (N-i) \log \left( 1 + \frac{x_i^\eta}{\beta + \tau_{i-1}^\eta} \right)} \] \quad (5.37)

After finding the MLE of $N$, $\beta$ the MLE of the parameter $\alpha$ is given by

\[ \alpha = \frac{n}{\sum_{i=1}^{n} (N-i) \log \left( 1 + \frac{x_i^\eta}{\beta + \tau_{i-1}^\eta} \right)} \] \quad (5.38)

5.3.5 Discussion and graphical description

The main interest in our application is to estimate some important software reliability measures of several SRGMs and analyze their predictability. All the selected models are studied as special cases of the Weibull and Burr Type XII models.

To make our study easier and faster we continued to refine the software toolset mentioned in Section 4.5. This software includes the required mathematical formulae of the Weibull and Burr Type XII models, and all the studied model's reliability measures are obtained as special cases. Also, we incorporate the software with another java software package (JFreeChart), in order to easy enter and present all the reliability quantities graphically.

Table 5.4 shows a summary of the data analysis. It contains the values of the estimates $\hat{N}$ and $\hat{\phi}$ which were computed by solving (5.33) and (5.34) simultaneously for the 38 DACS
and 20 DS4 data sets respectively. Also it contains the estimated number of remaining errors for each model.

Table 5.5 shows the estimated values of the model parameters and the estimated number of remaining errors of the three special cases of the common three parameter Burr model based on the same two data sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>DACS data</th>
<th>DS4 data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N$</td>
<td>$\hat{\phi}$</td>
</tr>
<tr>
<td>JM</td>
<td>40.63</td>
<td>2.8E-5</td>
</tr>
<tr>
<td>R</td>
<td>39.32</td>
<td>1.5E-8</td>
</tr>
<tr>
<td>W</td>
<td>38.88</td>
<td>4.4E-12</td>
</tr>
</tbody>
</table>

Table 5.4: Summary of data analyses based on three special cases of Weibull model using two failure data sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>DACS data</th>
<th>DS4 data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\hat{N}$</td>
<td>$\hat{\beta}$</td>
</tr>
<tr>
<td>L</td>
<td>44.67</td>
<td>196</td>
</tr>
<tr>
<td>SB</td>
<td>41.15</td>
<td>196902</td>
</tr>
<tr>
<td>THPB</td>
<td>40.10</td>
<td>96964664</td>
</tr>
</tbody>
</table>

Table 5.5: Summary of data analyses based on three special cases of Burr type XII model using two failure data sets.
As a result of substituting all the above calculated parameters' estimates in the mathematical expressions which are summarized in Table 5.2 and 5.3, we evaluate three important software reliability measures (the reliability function, MTTF, and failure rate). All these quantities are represented graphically as follows:

Firstly, the estimated software reliability, versus time, for the underlying models using DACS and DS4 data sets are given in Figures 5.2.a and 5.2.b respectively. From these Figures we can see the software reliability decreases with time, and in fact we may expect this to happen in the short-term; it may be caused by ineffective fixes or the introduction of new faults. But when n increases we expect the reliability to increase because of the decrease in the number of remaining errors as n increases, Figures 5.3.a and 5.4.a show that using DACS data (based on n=34 and n=38), while Figures 5.3.b and 5.4.b show the same thing using DS4 data (based on n=16 and n=20).

Secondly, the estimated median time to next failure MTTF are obtained in order to assess the reliability of the used data sets. The results of the estimated MTTF of the selected models using the DACS and DS4 data are shown in Figures 5.5.a and 5.5.b respectively. Figure 5.5.a shows that there is an agreement between most of models except SB and THPB models which show more optimistic prediction results. In Figure 5.5.b the median predictions show that there is a remarkable agreement between all the special cases of the Burr Type XII, and JM models, while R and W models give very optimistic results. For both data sets we can see that the L model gives the most pessimistic results. Also, the estimated MTTF of the two observed data sets increases over time, as shown in the two graphs.

Finally, the plots of the estimated failure rate functions of all the underlying models for DACS and DS4 data sets (after n=38 and 20 faults, respectively) are shown in Figures 5.6, 5.7, and 5.8. From these plots we observe:

- Figures 5.6.a and 5.6.b show that after eliminating 38 and 20 faults the estimated failure rate of JM model is stabilizing at 1.021E-4 and 0.028 for DACS and DS4 data sets, respectively. It should also be noted here that the failure rate of this model is
constant between the successive software failure and decreases when a fault is fixed after a failure is detected. From these two graphs we can see the increasing failure rate of R and W models as the time progress.

In Figures 5.7.a and 5.7.b we can notice that the estimated failure rate of the L model starts declining from the beginning, as most of the software reliability models assume. Here we assume that the probability of finding new failures becomes smaller and smaller as times goes on. This can be explained as, the earlier failures are easier to invoke than the later ones.

And lastly, Figures 5.8.a and 5.8.b show that the failure rate of the SB and THPB models show an initial rise then starts declining with time.
Figure 5.2: Plots of reliability functions based on some special cases of Weibull and Burr Type XII models using two failure data sets.

Figure 5.2.a: DACS data.

Figure 5.2.b: DS4 data.
Figure 5.3: Plots of reliability based on three special cases of Weibull model using two failure data sets.

Figure 5.3.a: DACS data.

Figure 5.3.b: DS4 data.
Figure 5.4: Plots of reliability based on three special cases of Burr Type XII model using two failure data sets.

Figure 5.4.a: DACS data.

Figure 5.4.b: DS4 data.
Figure 5.5: Median plots based on some special cases of Weibull and Burr Type XII models using two failure data sets.

**Figure 5.5.a: DACS data.**

**Figure 5.5.b: DS4 data.**
Figure 5.6: Failure rate plots of three special cases of Weibull model using two failure data sets.

Figure 5.6.a: DACS data.

Figure 5.6.b: DS4 data.
Figure 5.7.a: DACS data.

Figure 5.7.b: DS4 data.
Figure 5.8: Failure rate plots of SB and THPB models using two failure data sets.

**Figure 5.8.a**: DACS data.

**Figure 5.8.b**: DS4 data.
Now, we will begin evaluating the predictive accuracy of all the selected models based on the same aforementioned failure data sets. To do that, we will use all the three above-mentioned techniques. Firstly, we will begin with the comparison of the PLR technique, as we see in Table 5.6 this technique ranks the accuracy of all the underlying models. For the DACS data, the L model takes the first rank, with the R model second, while L, SB and THPB are superior to the other models for the DS4 data set.

Because of the inability of this technique to judge the model's accuracy objectively, we will investigate further by using the u-plot and y-plot techniques. The results of using those two techniques are summarized in Table 5.7. This table shows the Kolmogorov-Smirnov (KS) distance of both techniques for all the selected models using the above two failure data sets. For the DACS data, all the six models gave poor reliability predictions. At $\alpha=0.01$, all the deviations of the u-plots are very statistically significant from the line of unit slope. The L model performs the worst, with the THPB model second. Also, only SB and THPB models fit the DS4 data at $\alpha=0.01$, while the rest models show inaccurate reliability predictions.

The y-plot technique is necessary to tell us if the prediction errors (which may be present in the u-plot) are at least stationary. For the DACS data, only three models (JM, L and THSB) show significant y-plot results; their deviations from the unit slope line are statistically insignificant. All the selected models except W deviate insignificantly from the line of unit slope using DS4 data.

The results of the Prequential Likelihood Ratio (PLR) are presented graphically in Figure 5.9, whereas the u-plot and the y-plot results are presented graphically in Figures 5.10 and 5.11, respectively.
<table>
<thead>
<tr>
<th>Models</th>
<th>DACS data</th>
<th></th>
<th>DS4 data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n=38</td>
<td></td>
<td>n=20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOG(PLR)</td>
<td>Rank</td>
<td>LOG(PLR)</td>
<td>Rank</td>
</tr>
<tr>
<td></td>
<td>raw prediction</td>
<td>Against raw JM model</td>
<td>raw prediction</td>
<td>Against raw JM model</td>
</tr>
<tr>
<td><strong>Weibull model</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b = 1$</td>
<td>RM</td>
<td>RM</td>
<td>RM</td>
<td>RM</td>
</tr>
<tr>
<td>JM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b = 2$</td>
<td>11.135</td>
<td>$2^{nd}$</td>
<td>-8.478</td>
<td>$4^{th}$</td>
</tr>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b = 3$</td>
<td>-33.370</td>
<td>$3^{rd}$</td>
<td>-11.754</td>
<td>$5^{th}$</td>
</tr>
<tr>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Burr Type XII model</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\gamma = 1$</td>
<td>72.088</td>
<td>$1^{st}$</td>
<td>12.260</td>
<td>$1^{st}$</td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\gamma = 2$</td>
<td>-454.771</td>
<td>$4^{th}$</td>
<td>9.876</td>
<td>$2^{nd}$</td>
</tr>
<tr>
<td>SB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\gamma = 3$</td>
<td>-1757.478</td>
<td>$5^{th}$</td>
<td>-7.126</td>
<td>$3^{rd}$</td>
</tr>
<tr>
<td>THPB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.6: Predictive quality analysis results of two failure data sets using the PLR technique.
Table 5.7: Predictive quality analysis results of two profile data sets using the u-plot and the Y-plot techniques.

<table>
<thead>
<tr>
<th>Model</th>
<th>Type</th>
<th>Burst</th>
<th>Weibull</th>
<th>DHQ</th>
<th>HNQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>XI</td>
<td>«</td>
<td>0.314</td>
<td>0.272</td>
<td>0.243</td>
<td>0.557</td>
</tr>
<tr>
<td>XI</td>
<td>«</td>
<td>0.193</td>
<td>0.362</td>
<td>0.220</td>
<td>0.375</td>
</tr>
<tr>
<td>XI</td>
<td>«</td>
<td>0.249</td>
<td>0.444</td>
<td>0.248</td>
<td>0.376</td>
</tr>
<tr>
<td>XI</td>
<td>«</td>
<td>0.395</td>
<td>0.418</td>
<td>0.316</td>
<td>0.376</td>
</tr>
<tr>
<td>XI</td>
<td>«</td>
<td>0.336</td>
<td>0.414</td>
<td>0.333</td>
<td>0.375</td>
</tr>
</tbody>
</table>

The model fits the data set at α = 0.01.
The model does not fit the data set at α = 0.05.
Figure 5.9: The log (PLR) plots of raw predictions based on some special cases of Weibull and Burr Type XII models versus a reference model, the JM model using two failure data sets.

Figure 5.9.a: DACS data.

Figure 5.9.b: DS4 data.
Figure 5.10: kaw u-plots based on some special cases of weibull and burr type A11 models using two failure data sets.

Figure 5.10.a: DACS data.

Figure 5.10.b: DS4 data.
Figure 5.11: Karr y-plots based on some special cases of Weibull and Burr type XII models using two failure data sets.

Figure 5.11.a: DACS data.

Figure 5.11.b: DS4 data.
5.3.6 Conclusion and suggestions

The work on software reliability models started in the 1970's, the first model being presented in 1972. Since then, numerous software reliability models have been proposed with more models to assess the software quality being developed almost every year.

In this chapter we have modified the general formula of the JM model. As a result of that, a common three parameters Burr model, the general formula of the common well known Littlewood L model, has been obtained. The predictability of some special cases of the Burr type XII and Weibull models were investigated. Our investigation revealed the potential major problem of getting inaccurate raw prediction systems when using this conventional type of modeling.

Unfortunately, the introduction of new models of this type can be more detailed, complicated and yet cannot be guaranteed to predict accurately. Instead of that, it may be better to focus on reuse and refinement of the already existing SRGMs in a more productive way. Recently, the following refinement approaches have been presented to try to help in this case:

- The first approach, recalibration, can be used to get new better predictions by recalibrating the predictions obtained from the raw models. This approach was presented by Brocklehurst et al. (1990). The idea behind their approach is that the u-plot technique may help the user not only to select an accurate model, but also to dynamically develop the software reliability models so that a better predictive performance is achieved.

- In (1992) Lyu and Nikora proposed a different approach toward the software reliability measurement. Their opinion was that finding ways to reuse the existing models, may introduce better prediction result than introducing new model which can cope with project related information. As a result of their investigation a set of linear combination models are presented. These models have shown promising results when compared to the traditional single-model approaches.
We will continue our exploration of SRGMs in the next chapter by seeing to what extent these model refinement techniques really can help us.
Chapter 6

On Using Refinement Approaches for SRGM Prediction Improvement

6.1 Introduction

In Chapters 4 and 5 we studied many SRGMs but unfortunately, there was no single model that can be generally recommended to a specific software project. Some of the models gave good prediction results, with others performing poorly. However, none of them has proven to be accurate and trustworthy in all situations. These unsatisfactory prediction results motivate us to try some new refinement techniques in order to try to eliminate these difficulties.

In this Chapter we examine the effect of applying the two model refinement approaches (recalibration and model combination) mentioned at the end of the last chapter, to improving the prediction results produced by seven of the traditional SRGMs. For this purpose we use four observed data sets. Two of them are from Musa's data collection. These are high quality data sets; they were carefully collected and controlled to ensure their accuracy. The other two data sets are from a Philips development centre. These data sets were used to test the techniques in a setting that was more representative of normal industrial practice. Three
predictive validity analysis techniques are used to evaluate the prediction results before and after applying the two development approaches. The principle results of this work include:

- Generally, the development approaches enhance all the raw prediction systems but sometimes these improvements were still insignificant.
- The performance of the raw prediction systems affects the significance of the improvements that were obtained.
- In the case of the raw prediction systems, the recalibration approach worked better than the combination approach.
- Nevertheless, the combination approach helps to solve the possible difficulty of obtaining inaccurate recalibrated prediction systems.

This Chapter is organized as follows. Section 6.2 discusses, in some detail, the two model development approaches that we will apply in this Chapter in order to try to enhance the models' performance. An application, to examine the effectiveness of the refinement approaches is presented in Section 6.3. In this application we conduct four numerical examples using seven SRGMs and four real failure data sets, very detailed discussions are given and some useful results are concluded.

### 6.2 Refinement techniques

In this chapter, our main concern will centre upon developing some of the conventional SRGMs. In the following, we will describe briefly the two modification approaches that we are going to use to improve the predictive ability of the underlying models.

#### 6.2.1 The Recalibration approach

Usually, software reliability models are constantly biased in their predictions. Through measuring and analyzing the predictive accuracy of SRGMs, more accurate measures can be obtained by using some refinement approaches. Brocklehurst et al. (1990) illustrated in their paper the recalibration technique. This technique can be used to eliminate the bias, the
difference between the predicted and the true distribution, from the raw prediction system and construct a new improved prediction system. The recalibration technique represents the relation between the predicted $\hat{F}_i(t_i)$ and the true distribution $F_i(t_i)$ by the function $G_i$ where

$$F_i(t_i) = G_i[\hat{F}_i(t_i)]. \quad (6.1)$$

If we knew $G_i$ we could use it to adjust the raw prediction based on previous predictions. Practically, we do not know this function. But, the key notion in the recalibration approach is that in many cases the sequence $G_i$ is approximately stationary, i.e., $G_i = G$. This opens up the possibility of approximating $G_i$ with an estimate $G_i^*$ and so forming a new prediction:

$$\hat{F}_i^*(t_i) = G_i^*[\hat{F}_i(t_i)]. \quad (6.2)$$

In fact, there are two approaches to approximating the function $G_i$ with an estimate $G_i^*$. The first way suggests the use of the simple joined-up u-plot based upon the earlier predictions as a suitable estimator for the function $G_i$. But as we need the function $G_i^*$ to have the properties of continuity and differentiability, it is desirable to follow the second approach, which suggests the use of the spline technique to smooth the joined-up u-plot. This will be important when we come to examine the accuracy of reliability predictions via the PLR, since this requires continuous density predictions. Then the new prediction (6.2) recalibrates the raw model prediction, $\hat{F}_i(t_i)$ in the light of the knowledge of the accuracy of past predictions. In summary, any raw prediction system can be recalibrated by using the following four steps:

- Find the u-plot, $G_i^*$, depending upon the earlier predictions. It is desirable to use the spline-smoothed version to smooth the joined-up, step-function u-plot.
- Find $\hat{F}_i(t_i)$ by using the raw prediction system at stage $i$.
- Evaluate the recalibrated prediction system $\hat{F}_i^*(t_i) = G_i^*[\hat{F}_i(t_i)]$.
- To obtain a sequence of recalibrated predictions repeat at each stage $i$.  

180
6.2.2 The Combination approach

The model combination technique is based on combining several existing prediction systems to form one (hopefully) more accurate prediction system. Lyu and Nikora (1992) have presented four different ways to form a linear combination of models. Fundamentally, Lyu and Nikora formulated in their paper the following four combined models:

1) Equally-Weighted Linear Combined ELC model: This is the simplest combined model to create. Each component in this model has a constant, equal weight. In fact, the weight in this model is simply the arithmetic average of all the selected components.

2) Median-Oriented Linear Combined MLC model: this model selects the components whose predicted value lies between optimistic and pessimistic values. The justification for this approach is that the choice of median might be more moderate than the mean in some cases, since it can better tolerate an erroneous prediction which is far away from the others.

3) Unequally-Weighted Linear Combined ULC model: this model is similar to the MLC model except that instead of being only determined by the median value, the optimistic and pessimistic predictions make small contributions to the final prediction.

4) Dynamically-Weighted Linear Combined DLC model: the weights in this model are assumed to be changed according to changes in a model's applicability, since the applicability of any single model may change as testing progress. Here, the changes in prequential likelihood ratio that denotes a model's accumulated accuracy are used as the reference in assigning weights to each component.

Basically, to create a combined model the following steps can be adopted:

- Choose a basic set of models and separately apply each model to the failure data.
- Select the best performing models (the component models) by using the suitable predictive validity analysis techniques.
- Assign a weight to each selected model by using one of the methods that are described above.
- Then, form the combined prediction model for the final predictions.
In our study we will choose to formulate the DLC model. This combined model has shown refinements in predictive validity in several applications [Lyu and Nikora (1992), Minyan Lu et al. (2000)]. According to this approach we can define the combined model as follows:

The formula to obtain the pdf of this model is:

\[ f_i^s(t_i) = \sum_{s=1}^{r} w_i^s f_i^s(t_i) \]  

(6.3)

The cdf of this model is then given by:

\[ F_i^s(t_i) = \sum_{s=1}^{r} w_i^s F_i^s(t_i) \]  

(6.4)

Finally, we assign the weight to each selected model \( j \) as follows:

\[ w_i^s = \frac{PL_i^s}{\sum_{s=1}^{r} PL_i^s}, \quad s=1,2,\ldots,r \]  

(6.5)

where \( PL_i^s = \prod_{k=j}^{i} f_k^s(t_k), \quad i = j, j+1, \ldots, n \), \( j \) is a number sufficiently large for the first prediction, \( n \) is the total number of software failures observed from the selected software, \( \sum_{s=1}^{r} w_i^s = 1 \) and \( r \) is the number of the selected models.

The combined model tends to preserve the features inherited from its component models. Also, because each component performs reliability calculations independently, the combined model remains fairly simple. The component models are plugged into the combined model only at the last stage for final predictions. Selecting appropriate component models is, of course, important to the success of the combination model.
In our study we apply this technique twice. First we will use it over the raw models. We will denote the resulting model the C model. We will then apply it over the recalibrated models and this time we use the symbol CR to denote the resulting model.

6.3 Application of the refinement techniques

In what follows we evaluate the performance of seven conventional SRGMs before and after applying the two refinement approaches.

6.3.1 Data sets

In our evaluation we used four data sets. Tables 6.1 and 6.2 display the first two data sets. These two data sets were collected from a Philips development centre. The first one consists of 246 inter-failure times and the second one consists of 312 inter-failure times. The failure data sets in Table 6.3 and 6.4 are from J. Musa's (1979) “Software Reliability Data”, available from DACS, Rome Air Development Center, New York. Figures [(6.1.a)-(6.1.d)] show the failure times in sequence of all these four data sets. The improvement trend is visible in Figures 6.1.a, 6.1.c, and 6.1.d, while Figure 6.1.b shows no improvement trend. We should note this raises a concern about the process of reliability growth modeling. Conventionally, SRGMs are applied during a testing phase that is assumed to take place after the major part of software development has taken place. In practice, as here, we often find incremental enhancements are made to the overall system as testing continues as a parallel activity.
Table 6.1: Philips failure data 1: execution times between successive failures in minutes; read from left to right, number of failures = 246.

| 43 | 3  | 0.5  | 8.5 | 7     | 0.3333 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 12.5 |
| 1  | 0.5 | 0.0417 | 6.4583 | 0.4167 | 0.0833 | 0.5  | 0.4167 | 0.0833 | 0.5  | 5     |
| 0.3333 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 1.5 | 0.4167 | 0.0833 | 2.5  | 0.5  | 0.5  | 0.4167 | 0.0833 | 0.5  | 0.4167 | 0.0833 | 0.5  | 3     | 0.3333 |
| 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |
| 0.5  | 0.5  | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 0.0417 | 1     | 0.4167 | 0.0417 | 0.0417 |

Table 6.2: Philips failure data 2: execution times between successive failures in minutes; read from left to right, number of failures = 312.
Table 6.3: Musa SS1A data: execution times between successive failures in seconds; read from left to right, number of failures = 112.

Table 6.4: Musa SS1C data: execution times between successive failures in seconds; read from left to right, number of failures = 277.
Figure 6.1: Failures times in order of occurrence for several real data sets.

Figure 6.1.a: Data 1 of Table 6.1.

Figure 6.1.b: Data 2 of Table 6.2.
Figure 6.1.c: Data 3 of Table 6.3.

Figure 6.1.d: Data 4 of Table 6.4.
6.3.2 Conventional models used

We focused on the following seven of the conventional SRGMs in our empirical evaluations:

1) Jelinski-Moranda, JM, model.
2) R model.
3) W model.
4) Littlewood, L, model.
5) SB model.
6) THPB model.
7) Duane, Du, model.

The first six models are studied as special cases of the two general models (Weibull and Burr Type XII). By giving the parameter \( b \) in equation (5.20) the value 1, 2, and 3 we obtain (JM, R, and W) models respectively, while (L, SB, and THPB) models can be obtained by giving the parameter \( \gamma_2 \) in equation (5.30) the values 1, 2, and 3 respectively. As discussed, the main advantage of this is to avoid duplication of effort and time caused by studying each model separately. The last one, Duane model, was derived from the hardware reliability area. Duane has the property of converging over time to infinite failure intensity. Thus, if this model provides a good fit to the data, it offers a warning that a significant fault finding activity still needs to be performed; the number of residual faults in the software product is not stabilizing. The Duane model is a two-parameter infinite failures model. Unlike the finite failure models, the parameters \( a \) and \( b \) cannot be related to any physical characteristics of the system. Interestingly, or perhaps disappointingly for those concerned, in our experience the Duane model does provide a good fit to certain data sets. More descriptions of these models were provided in Chapters 4 and 5.

6.3.3 Techniques for the analysis predictive accuracy

In our application we will use the following techniques to analyze the predictive validity of the underlying models:

a) The u-plot.
6.3.3 Discussion of results

Our main focus in this study is to improve the predictive ability of some conventional software reliability growth models. To accomplish that, two refinement approaches are used. Some predictive validity analysis techniques are also used to examine the performance of these models before and after applying the refinement approaches. As with all the earlier work in this thesis, we used our software tools to perform the experiments. In more detail, our software helps to perform the following functions:

1) Derive the estimates of the unknown parameters that are involved in all the seven models we investigate using the four above mentioned interfailure time data sets. This are carried out by using the maximum likelihood estimation method. All the estimates based on \( (t_1, t_2, ..., t_j, ...) \) for \( j = 20, 21, ..., n \), where \( n \) takes the values (246, 312, 112, 277) for each data sets correspondingly.

2) Evaluate the initial raw predictions \( \hat{F}_j(t_j) \) for \( j = 20, 21, ..., n \), by substituting all the above computed estimates into suitable mathematical expressions derived from the seven underlying models [equations (5.22), (5.31), and (4.73) respectively]. As a result, 28 raw prediction systems were obtained in our study.

3) Analyze the predictive validity of all the 28 raw prediction systems on all four data sets by applying all the above mentioned predictive validity analysis techniques. We start with the PLR technique, which helps to rank the accuracy of all the underlying models. Then, investigate more by using the u-plot and the y-plot techniques. The prediction system is considered inaccurate if the
deviation of the u-plot is statistically significant at the 10 percent level. At the same level of significance, any significant departure in the y-plot will indicate that there is no constant bias for this model. After that, all the inaccurate raw prediction results will be applicants for the refinement techniques that we mentioned previously.

1) To improve the predictive validity of all the poor raw prediction systems we previously obtained, the program utilizes the recalibration approach by performing the following steps:

- Finding the parametric spline function $G_i^*$, for $i = 40, 41, ..., n$ based on the u-plot (using $\hat{P}_j(t_j)$ for $j = 20, 21, ..., i-1$).

- Then, substitute both the function $G_i^*$ and the corresponding raw prediction $\hat{P}_i(t_i)$ in equation (6.2) to obtain the recalibrated predictions $P_i^*(t_i)$.

- As a result of applying this enhancement technique, (207, 273, 73, 238) recalibrated data points were obtained from each one of the seven models we investigated (i.e. we obtained 28 recalibrated prediction systems).

2) As a second way to try to enhance the inaccurate prediction systems, the program implements the combination approach. In our investigation this approach was used twice. Firstly, (as with the recalibration approach) over the inaccurate raw predictions and subsequently over the inaccurate recalibrated predictions (in order to try to solve this problem). To create the combined model the following points are needed:

- Firstly, select the four best performing models among the raw (or the recalibrated) ones by using the above mentioned predictive validity analysis techniques (the PLR, u-plot, and y-plot).

- Secondly, assign weights to each selected model to construct the DLC model as described above in Section 6.2.2.

- Then, form the final combined prediction systems $F_i^*(t_i)$, for $i = 40, 41, ..., n$. 

190
In our experiments we applied this approach 8 times. The first four times used the raw prediction systems, while we created the other four combined models from the recalibrated prediction systems.

3) And in the final step, the program uses the same tools as in item 3 above (the PLR, the u-plot and the y-plot), to evaluate the accuracy of both the recalibrated and the combined models.

4) In order to easily compare and study all the obtained numerical results the program, we produce graphs for:
   - The raw data plotted as the successive failure times against the failure number.
   - The results we obtained from applying all the predictive validity analysis techniques.

6.3.4.1 Analysis of Data 1

To make a comparison between all the underlying models, we used the PLR technique. In Figure 6.2.a the log (PLR) is plotted for all the underlying models against a reference model, the raw Du model. This graph shows L and SB models are superior to the other five raw models. The graph shows also that the combined C model performs the best overall according to this technique.

For further investigation about the predictive accuracy of these models the u-plot and y-plot techniques were used. The u-plots, in Figure 6.2.b, show that all the seven models give poor prediction results. The Du and L models are superior to the other five models, although still not providing acceptable predictive accuracy. In addition, all the significant deviations from the line of unit slope, as observed by the y-plots in Figure 6.2.c, indicate that the prediction errors for these models are not consistent. From these two graphs, we can see the effect of applying the combination method over the raw prediction systems, as well. The C model performs the best, although its u- and y-plots also deviate significantly from the line of unit slope.
Because of the above poor performance of all the underlying raw models, we can not really trust any of them. Therefore, all these raw models are candidates for recalibration. The u-plots of the recalibrated prediction, in Figure 6.2.d, are much better than the above u-plots of the raw predictions, but still the deviations from the line of unit slope are statistically significant for most of the recalibrated models. Only three recalibrated models (R, W and Du) show no significant deviations from the line of unit slope. The same thing can be seen by the y-plots, in Figure 6.2.e. Another thing we can see from these two figures, the plot of CR model differs insignificantly from the unit slope line.

In Figure 6.2.f the log (PLR) is plotted again, but this time to compare the predictive accuracy of the enhanced models. In this case we used the recalibrated Du as a reference model. This graph indicates that the SB and L models are superior to the other five models. In order to include the C and CR models in the comparison, the log (PLR) is plotted for these two models in the same figure. These two models show a consistent trend and take the rank third and first respectively (see Table 6.5).

Finally, Figure 6.2.g shows the log (PLR) plots of the recalibrated versus the raw predictions. There is an enhancement in the predictive validity in all cases. The best enhancement occurs with THSB first, with SB second.

According to the validity analysis of this data set, we can conclude that all the used models were improved after recalibration. However, for some of these models the improvement is still relatively insignificant. The combination method also improved the raw predictions but this method only gave significant result with the recalibrated prediction systems. Hence the models that we recommend here are the R and CR models.

In conclusion, according to the above predictive validity analysis methods the four best performing raw models here are the L, SB, R, and Du models, although their performance is still not good. These four models were then used to form the C model, as described in Section 6.2. Also we used the recalibrated Du, W, R, and THPB models to create the CR model.
Figure 6.2: A refinement example using Data 1 of Table 6.1.

Figure 6.2.a: The log(PLR) plots of raw and combined prediction systems versus a reference model, the raw Du model for Data 1.

Figure 6.2.b: Raw and combined u-plots for Data 1.
Figure 6.2.c: Raw and combined y-plots for Data 1.

Figure 6.2.d: Recalibrated and combined u-plots for Data 1.
Figure 6.2.e: Recalibrated and combined y-plots for Data 1.

Figure 6.2.f: The log(PLR) plots of recalibrated and combined prediction systems versus a reference model, the recalibrated Du model for Data 1.
6.3.4.2 Analysis of Data 2

The comparison using the PLR, together with the investigation by the u-plot and y-plot techniques give an indication of whether a model has a high predictive ability or not. In our second example we will also begin by plotting the log (PLR) of all the raw models against the raw Du model. The resulting plots are shown in Figure 6.3.a. This graph reveals that L performs the second best, with SB third. In addition to that, the graph shows that the trend of the C model is so consistent, and with respect to this technique this model ranks the first as seen by Table 6.5.

The u-plots in Figure 6.3.b show again that all the raw predictions are extremely inaccurate. The most inaccurate prediction results come from JM model, with W and Du models second and third, respectively. Also, insignificant raw prediction results are obtained by the y-plots in Figure 6.3.c, which emphasizes that we cannot depend on these raw
prediction systems on this second data set. As a result, all these raw prediction systems need to be improved. The C model gives much better prediction results but still we can not accept this model according to its u-plot and y-plot significant KS distance.

We again used the recalibration technique in order to get better predictive validity. Figures 6.3.d and 6.3.e show the u-plots and the y-plots, respectively, of all the enhanced models. After recalibration, we can see that all the deviations from the unit slope of the u-plots are still statistically significant. The y-plots give also significant deviations for four of the studied models. So even though there is a considerable improvement in all the raw prediction systems, these recalibrated prediction systems still cannot be accepted. Here, in addition to the recalibration approach and as a way to solve the problem of insufficient predictive validity of the recalibrated prediction systems, the combination technique was used. And as a result, we can see from the plot of the CR model in the same above two figures a very significant result. So we can say that in this case recalibration associated with combination again leads to the best prediction.

In Figure 6.3.f the PLR analysis against the recalibrated Du shows a steady increase in all cases except for the THSB and SB models which ended with a little decrease. This analysis indicates that although the recalibration technique did not give significant results with all the raw studied models, a great improvement has occurred in the all these raw models. The C and CR take the ranks fifth and third, correspondingly.

Figure 6.3.g also give us extra evidence that recalibration can improve all the raw prediction systems. The greatest improvement is made to THSB and SB models, while the L model gets the least improvement.

To summarize the analysis of the second data set, we can say that none of the modification techniques works well alone and gives significant results with the raw prediction systems. This may be because this data set gives extremely poor raw prediction results. Only with the recalibrated predictions does the combination approach work well, so the CR is the only model we advise for use with this data set.

For this data set we created the C model by using the raw L, SB, THPB, and R models, whereas the CR model were formed by using the recalibrated JM, THPB, SB, and L models.
Figure 6.3: A refinement example using Data 2 of Table 6.2.

Figure 6.3.a: The log(PLR) plots of recalibrated and combined prediction systems versus a reference model, the recalibrated Du model for Data 2.

Figure 6.3.b: Raw and combined u-plots for Data 2.
Figure 6.3.c: Raw and combined y-plots for Data 2.

Figure 6.3.d: Recalibrated and combined u-plots for Data 2.
Figure 6.3.e: Recalibrated and combined y-plots for Data 2.

Figure 6.3.f: The log(PLR) plots of recalibrated and combined prediction systems versus a reference model, the recalibrated Du model for Data 2.
6.3.4.3 Analysis of Data 3

Once again, with this third data set we start our analysis by comparing the predictive accuracy of all the raw studied prediction systems. Figure 6.4.a shows the results of this analysis. The THSB and SB models are the worst overall. Once more, we can see here the best result is coming from the C model.

In Figure 6.4.b, the u-plots again show that all the original seven models perform poorly. But we can notice that the raw prediction results of this data set are better than the raw prediction results of the above two data sets. The worst raw prediction results are coming from JM model. Also, in Figure 6.4.c we can see better prediction results by the y-plots than we obtained before with the first and second data sets. This figure shows no significant
deviation from the line of unit slope for some of the models. But still these raw prediction systems are not good enough and none of them can be relied upon. Therefore, the enhancement techniques need to be applied here. These two figures show us, for the first time in our study, the combination method works well with the raw prediction systems and give significantly improved prediction results.

Figures 6.4.d and 6.4.e shows the effect of recalibration on the raw prediction systems. The recalibration improved all the raw prediction systems; all the models show insignificant deviation from the line of unit slope. And, of course we expect to see very significant prediction results from the CR model in this case.

We will use the PLR analysis to compare the predictive accuracy of all the recalibrated models and also the combined model. For this analysis, as before, the recalibrated Du is the reference model. After applying the modification methods, Figure 6.4.f shows that the best prediction results are coming from the CR model, with the C and L models next best.

Lastly, Figure 6.4.g confirms that the recalibrated predictions of all the seven models are superior to the raw ones.

In conclusion, for this data set, the two enhancement methods work very well and all the selected raw prediction systems show dramatic improvement. So all the developed models can be recommended for this data set, specifically the CR, C, and Du models.

Here, we have chosen the raw L, R, Du, and W models to create the C model, while the recalibrated L, R, W, and JM models were chosen to form the CR model.
Figure 6.4: A refinement example using Data 3 of Table 6.3.

Figure 6.4.a: The log(PLR) plots of recalibrated and combined prediction systems versus a reference model, the recalibrated Du model for Data 3.

Figure 6.4.b: Raw and combined u-plots for Data 3.
Figure 6.4.c: Raw and combined y-plots for Data 3.

Figure 6.4.d: Recalibrated and combined u-plots for Data 3.
Figure 6.4.e: Recalibrated and combined y-plots for Data 3.

Figure 6.4.f: The log(PLR) plots of recalibrated and combined prediction systems versus a reference model, the recalibrated Du model for Data 3.
6.3.4.4 Analysis of Data 4

First, we will begin with the comparison of the PLR technique. Figure 6.5.a shows the PLR analysis against the raw Du model. With respect to this analysis we can see the C model takes the first rank, with the L model second. But this analysis is not sufficient and tells us nothing about the difference between the predicted and the true reliability. For this reason, in the following we will proceed by using the u- and y-plot techniques.

Once again, the u-plots of this data set in Figure 6.5.b indicate inaccurate raw prediction results. But also these raw results are relatively better than the ones that were obtained by the first two data sets. The worst results are coming from the Du and THPB models, respectively. The y-plots in Figure 6.5.c also present insignificant results except for the JM and Duane models. The u-plot of the C model shows significant deviation from the line of unit slope. So

Figure 6.4.g: The log(PLR) plots of recalibrated versus raw prediction systems for Data 3.
even with its significant y-plot result, this indicates that the combination method does not work well with these raw prediction systems.

Again, further help may come from the previously mentioned enhancement tools. The recalibrated results of all the seven models are shown in Figures 6.5.d and 6.5.e. No significant deviation from the line of unit slope can be seen in these two graphs. This reveals that the recalibration technique improved the predictive accuracy of these raw models significantly. And for sure, applying the combination method over these recalibrated predictions will give significant results, as the CR model reveals in the same two graphs.

Figure 6.5.f will help us to choose the best performing models after using the enhancement techniques. From this figure we can see that the CR model performs the best, with the L and Du second and third best, respectively.

And finally, Figure 6.5.g will help us to judge the effect of the recalibration technique on this data set. According to this figure we can see that the Du, W and THSB models showed the greatest enhancement. This may be because those three models had the worst raw predictions, so they were really in greatest need of improvement.

By the end of our analysis of this last data set, we can say that a very significant enhancement has been shown by using the recalibration method. But on the other hand, the combination approach did not give significant results with the selected raw prediction systems.

The components of the C model are the raw JM, R, L, and W models, and the components of the CR model are the recalibrated L, Du, W, R models.

Table 6.5 summarizes the results of the PLR technique on all the above data sets. According to this technique we rank the predictive validity of all the studied models.

To measure the degree of deviation of u-plot and y-plot from the line of unit slope before and after applying the refinement techniques we use the Kolmogorov Smirnov, KS, distance, as previously mentioned. These distances are evaluated and summarized in Table 6.6.
Figure 6.5: A refinement example using Data 4 of Table 6.4.

Figure 6.5.a: The log(PLR) plots of raw and combined prediction systems versus a reference model, the raw Du model for Data4.

Figure 6.5.b: Raw and combined u-plots for Data 4.
Figure 6.5.c: Raw and combined y-plots for Data 4.

Figure 6.5.d: Recalibrated and combined u-plots for Data 4.
Figure 6.5.e: Recalibrated and combined y-plots for Data 4.

Figure 6.5.f: The log(PLR) plots of recalibrated and combined prediction systems versus a reference model, the recalibrated Du model for Data 4.
Figure 6.5.g: The log(PLR) plots of recalibrated versus raw prediction systems for Data 4.
<table>
<thead>
<tr>
<th>Data sets</th>
<th>Models</th>
<th>LOG(PLR) raw prediction against raw Duane model</th>
<th>Rank</th>
<th>LOG(PLR) Recalibrated and combined prediction against recalibrated Duane model</th>
<th>Rank</th>
<th>LOG(PLR) recalibrated prediction against raw prediction</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weibull model</td>
<td>$\eta = 1$ JM</td>
<td>366.586</td>
<td>5th</td>
<td>238.070</td>
<td>6th</td>
<td>758.514</td>
<td>4th</td>
</tr>
<tr>
<td></td>
<td>$\eta = 2$ R</td>
<td>468.125</td>
<td>4th</td>
<td>210.504</td>
<td>7th</td>
<td>629.409</td>
<td>6th</td>
</tr>
<tr>
<td></td>
<td>$\eta = 3$ W</td>
<td>210.164</td>
<td>6th</td>
<td>29.832</td>
<td>8th</td>
<td>706.698</td>
<td>5th</td>
</tr>
<tr>
<td>Burr Type XII model</td>
<td>$\gamma = 1$ L</td>
<td>1151.397</td>
<td>2nd</td>
<td>529.642</td>
<td>4th</td>
<td>265.275</td>
<td>7th</td>
</tr>
<tr>
<td></td>
<td>$\gamma = 2$ SB</td>
<td>691.487</td>
<td>3rd</td>
<td>695.388</td>
<td>2nd</td>
<td>890.931</td>
<td>2nd</td>
</tr>
<tr>
<td></td>
<td>$\gamma = 3$ THPB</td>
<td>-971.336</td>
<td>7th</td>
<td>319.758</td>
<td>5th</td>
<td>2178.124</td>
<td>1st</td>
</tr>
<tr>
<td>Du model</td>
<td>RM</td>
<td>1439.886</td>
<td>1st</td>
<td>552.855</td>
<td>3rd</td>
<td>887.030</td>
<td>3rd</td>
</tr>
<tr>
<td>C model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data 1 n=246</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weibull model</td>
<td>$\eta = 1$ JM</td>
<td>538.125</td>
<td>6th</td>
<td>479.113</td>
<td>6th</td>
<td>1117.266</td>
<td>5th</td>
</tr>
<tr>
<td></td>
<td>$\eta = 2$ R</td>
<td>563.191</td>
<td>5th</td>
<td>325.584</td>
<td>7th</td>
<td>938.671</td>
<td>6th</td>
</tr>
<tr>
<td></td>
<td>$\eta = 3$ W</td>
<td>251.471</td>
<td>7th</td>
<td>222.013</td>
<td>8th</td>
<td>1146.821</td>
<td>4th</td>
</tr>
<tr>
<td>Burr Type XII model</td>
<td>$\gamma = 1$ L</td>
<td>1704.619</td>
<td>1st</td>
<td>939.410</td>
<td>4th</td>
<td>411.070</td>
<td>7th</td>
</tr>
<tr>
<td></td>
<td>$\gamma = 2$ SB</td>
<td>1580.879</td>
<td>2nd</td>
<td>1668.369</td>
<td>2nd</td>
<td>1263.768</td>
<td>2nd</td>
</tr>
<tr>
<td></td>
<td>$\gamma = 3$ THPB</td>
<td>1055.542</td>
<td>4th</td>
<td>2058.338</td>
<td>1st</td>
<td>2179.074</td>
<td>1st</td>
</tr>
<tr>
<td>Du model</td>
<td>RM</td>
<td>1910.357</td>
<td>3rd</td>
<td>734.079</td>
<td>5th</td>
<td>1176.279</td>
<td>3rd</td>
</tr>
<tr>
<td>C model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.5: Predictive quality analysis of raw, recalibrated and combined Predictions of four real data sets using the PLR technique.
<table>
<thead>
<tr>
<th>Data sets</th>
<th>Models</th>
<th>LOG(PLR) raw prediction Against raw Duane model</th>
<th>Rank</th>
<th>LOG(PLR) recalibrated prediction against recalibrated Duane model</th>
<th>Rank</th>
<th>LOG(PLR) recalibrated prediction against raw prediction</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Weibull model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\eta=1$ JM</td>
<td>-59.765</td>
<td>4$^{th}$</td>
<td>-335.999</td>
<td>6$^{th}$</td>
<td>95.860</td>
<td>5$^{th}$</td>
</tr>
<tr>
<td></td>
<td>$\eta=2$ R</td>
<td>-45.394</td>
<td>3$^{rd}$</td>
<td>-330.999</td>
<td>4$^{th}$</td>
<td>86.489</td>
<td>6$^{th}$</td>
</tr>
<tr>
<td></td>
<td>$\eta=3$ W</td>
<td>-96.088</td>
<td>5$^{th}$</td>
<td>-335.894</td>
<td>5$^{th}$</td>
<td>132.288</td>
<td>4$^{th}$</td>
</tr>
<tr>
<td>Data3  n=112</td>
<td>Burr Type XII model</td>
<td>$\gamma=1$ L</td>
<td>2$^{nd}$</td>
<td>81.888</td>
<td>3$^{rd}$</td>
<td>30.504</td>
<td>7$^{th}$</td>
</tr>
<tr>
<td></td>
<td>$\gamma=2$ SB</td>
<td>-4690.791</td>
<td>6$^{th}$</td>
<td>-4713.974</td>
<td>7$^{th}$</td>
<td>348.911</td>
<td>3$^{rd}$</td>
</tr>
<tr>
<td></td>
<td>$\gamma=3$ THPB</td>
<td>-10224.134</td>
<td>7$^{th}$</td>
<td>-10040.765</td>
<td>8$^{th}$</td>
<td>556.383</td>
<td>1$^{st}$</td>
</tr>
<tr>
<td></td>
<td>Du model</td>
<td>RM</td>
<td>RM</td>
<td>RM</td>
<td>RM</td>
<td>372.094</td>
<td>2$^{nd}$</td>
</tr>
<tr>
<td></td>
<td>C model</td>
<td>846.271</td>
<td>1$^{st}$</td>
<td>474.177</td>
<td>2$^{nd}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CR model</td>
<td></td>
<td></td>
<td>480.628</td>
<td>1$^{st}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data 4  n=277</td>
<td>Weibull model</td>
<td>$\eta=1$ JM</td>
<td>3$^{rd}$</td>
<td>-1103.308</td>
<td>6$^{th}$</td>
<td>112.286</td>
<td>6$^{th}$</td>
</tr>
<tr>
<td></td>
<td>$\eta=2$ R</td>
<td>-460.175</td>
<td>4$^{th}$</td>
<td>-928.753</td>
<td>4$^{th}$</td>
<td>475.742</td>
<td>5$^{th}$</td>
</tr>
<tr>
<td></td>
<td>$\eta=3$ W</td>
<td>-741.625</td>
<td>5$^{th}$</td>
<td>-860.691</td>
<td>3$^{rd}$</td>
<td>825.253</td>
<td>2$^{rd}$</td>
</tr>
<tr>
<td></td>
<td>Burr Type XII model</td>
<td>$\gamma=1$ L</td>
<td>2$^{nd}$</td>
<td>260.265</td>
<td>2$^{nd}$</td>
<td>69.254</td>
<td>7$^{th}$</td>
</tr>
<tr>
<td></td>
<td>$\gamma=2$ SB</td>
<td>-21654.163</td>
<td>7$^{th}$</td>
<td>-22119.217</td>
<td>8$^{th}$</td>
<td>482.326</td>
<td>4$^{th}$</td>
</tr>
<tr>
<td></td>
<td>$\gamma=3$ THPB</td>
<td>-16292.974</td>
<td>6$^{th}$</td>
<td>-16724.782</td>
<td>7$^{th}$</td>
<td>617.494</td>
<td>3$^{rd}$</td>
</tr>
<tr>
<td></td>
<td>Du model</td>
<td>RM</td>
<td>RM</td>
<td>RM</td>
<td>RM</td>
<td>944.319</td>
<td>1$^{st}$</td>
</tr>
<tr>
<td></td>
<td>C model</td>
<td>2533.022</td>
<td>1$^{st}$</td>
<td>-998.894</td>
<td>5$^{th}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CR model</td>
<td></td>
<td></td>
<td>1402.106</td>
<td>1$^{st}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.5: Continued.
<table>
<thead>
<tr>
<th>Data sets</th>
<th>Models</th>
<th>Raw predictions</th>
<th>Recalibrated Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>KS (u-plot)</td>
<td>Status</td>
</tr>
<tr>
<td>Data 1</td>
<td>Weibull model</td>
<td>$\eta=1$ JM</td>
<td>0.744</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\eta=2$ R</td>
<td>0.559</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\eta=3$ W</td>
<td>0.704</td>
</tr>
<tr>
<td></td>
<td>Burr Type XII model</td>
<td>$\gamma=1$ L</td>
<td>0.321</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\gamma=2$ SB</td>
<td>0.430</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\gamma=3$ THPB</td>
<td>0.494</td>
</tr>
<tr>
<td></td>
<td>Du model</td>
<td>0.173</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>C model</td>
<td>0.135</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>CR model</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Data 2</td>
<td>Weibull model</td>
<td>$\eta=1$ JM</td>
<td>0.967</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\eta=2$ R</td>
<td>0.720</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\eta=3$ W</td>
<td>0.841</td>
</tr>
<tr>
<td></td>
<td>Burr Type XII model</td>
<td>$\gamma=1$ L</td>
<td>0.451</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\gamma=2$ SB</td>
<td>0.534</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\gamma=3$ THPB</td>
<td>0.555</td>
</tr>
<tr>
<td></td>
<td>Du model</td>
<td>0.755</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>C model</td>
<td>0.244</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>CR model</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6.6: Predictive quality analysis of raw, recalibrated and combined predictions of four real data sets using the u-plot and the y-plot techniques.
<table>
<thead>
<tr>
<th>Data sets</th>
<th>Models</th>
<th>Raw predictions</th>
<th>Recalibrated Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>KS (u-plot)</td>
<td>Status</td>
</tr>
<tr>
<td>Data 3</td>
<td>Weibull model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=112</td>
<td>$\eta=1$ JM</td>
<td>0.618</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\eta=2$ R</td>
<td>0.262</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\eta=3$ W</td>
<td>0.433</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Burr Type XII model</td>
<td>0.349</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\gamma=1$ L</td>
<td>0.298</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\gamma=2$ SB</td>
<td>0.495</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Du model</td>
<td>0.407</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>C model</td>
<td>0.131</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>CR model</td>
<td>0.051</td>
<td>*</td>
</tr>
<tr>
<td>Data 4</td>
<td>Weibull model</td>
<td>0.155</td>
<td>-</td>
</tr>
<tr>
<td>n=277</td>
<td>$\eta=1$ JM</td>
<td>0.363</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\eta=2$ R</td>
<td>0.513</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\eta=3$ W</td>
<td>0.448</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Burr Type XII model</td>
<td>0.436</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\gamma=1$ L</td>
<td>0.632</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Du model</td>
<td>0.758</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>C model</td>
<td>0.164</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>CR model</td>
<td>0.051</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 6.6: Continued.
6.3.5 Conclusions

Conventional approaches to software reliability modeling allow the user to formulate predictions using data from the software testing process. However, it is necessary to consider the potential threats to validity of the prediction systems associated with using this type of modeling. In this chapter we have discussed this difficult problem, which still provides a barrier to the more extensive adoption of reliability modeling. Some causes and some suggested solutions have been stated. For the purpose of enhancing the predictive validity of the studied models, two refinement approaches, recalibration and model combination, have been studied. Essentially, our study combined with other studies [Brocklehurst et al. (1990), Brocklehurst and Littlewood (1992), Lyu and Nikora (1992)] has shown that generally these development approaches did help to improve this type of conventional software reliability model. However, in our investigation we have also noticed that, some times and particularly with the extremely poor performance prediction systems, the improvements are still insignificant. In other words, some of the poor prediction systems could not benefit from using these modified approaches. Also, and as a useful result, we have seen that even though the combination approach did not work as well as the recalibration approach in the case of the raw prediction systems, this approach did help to improve the validity of the prediction results after applying the recalibration technique.
Chapter 7

Conclusion and Suggestions for Future Research

7.1 Conclusions

As our dependence on software is growing more and more, the need for reliable software also increases. Over the past thirty-five years, the assessment of software reliability has become very important because of our increasing reliance on software. As we have seen, many software reliability models have been proposed in the literature, and yet there is still no convincing single solution to this problem.

There are several different ways to model a software failure process. One class of these models is that of the NHPP model. The best model is the one that predicts the future stochastic behavior of the system failure process better than the others. In our thesis we tested the performance of eleven NHPP models numerically and graphically. Our objective is to see how well our selected models fit four independent failure data sets and try to identify the preferred model for reliability prediction in each case. The graphical representations of our study, using the reliability predictions, demonstrated that:

- There is an inherent noise in the test data so it is impossible to find a single model that is guaranteed to give a reliability prediction with a high level of confidence.
Different judgments about the quality of the software could be made, depending on the choice of the model. So there is a need for further objective investigation.

Numerically, the predictability of the selected models was investigated by using four different objective Goodness-of-fit techniques; AIC, KS, $\chi^2$ and SSE. According to this investigation, we concluded the following points:

- There is commonly great disagreement in predictions, while no one SRGM is adequately more trustworthy than others in terms of predictive quality in all applications. Each SRGM works well with a specific case but no single model provides a best fit in all cases. Actually, similar results have been concluded by several studies e.g. [Abdel-Ghaly et al. (1986), Khoshgoftaar and Woodcock (1991), and Brocklehurst et al. (1990)].

- Different model selection methods indicated different predictive qualities of the same SRGM by using the same failure data set.

- The extreme variation in the reliability prediction from some of the models may indicate the need to gain more information before making a release decision for some of the projects.

- Even though we cannot choose a preferred model for all cases, each model can provide some useful information about the project under test.

- The complexity of the model’s assumptions affects its fitness to certain data sets. These assumptions can limit the applicability and the effectiveness of those models.

- The fitness of the Duane model to a specific data set can provide a useful warning that this data set still needs to be tested more before making a release decision.

One of the well known SRGMs, the Littlewood model was generalized. We provided theoretical and mathematical foundations to support our generalization. In order to support the
study of several SRGMs, which is preferable when studying this type of modeling, consuming less effort and time, we considered several special cases of the proposed model, the Burr Type XII, and the general Weibull model. Three model selection techniques were used to validate the underlying SRGMs. Our study demonstrated that inaccurate predictive validity can be obtained when using this conventional type of modeling. This result then motivated us to use some enhancement approaches.

In order to try to improve the accuracy of predictions when using this conventional type of modeling we applied two improvement approaches, recalibration and model combination. These two approaches share the idea of reuse of the original prediction systems, in a way that analyzes the test data in a more productive way. In order to investigate the effectiveness of the two approaches, we conducted evaluations using actual software failure data. Our examples were based on seven SRGMs. Furthermore, three techniques were applied to examine the performance of the underlying models before and after applying the two development approaches. Our study has shown that although these enhancement approaches can work well, they cannot be guaranteed to produce reliable prediction results all the time.

In practice, all the extant reliability growth models show a wide variability in predictive accuracy across a variety of data sets. There are a number of reasons for this. According to our investigations we summarize several of these problems and some corresponding solutions:

- **The lack of experience:** users who do not have enough prior information to choose a model from so many existing models tend to choose a model (or models) blindly. They often apply models even when their underlying assumptions are not valid. Also, there is a natural tendency to select simple models, partly because they are more easily dealt with and partly because in many situations insufficient data is available to support more complex methods. Disregarding the complexity of a system may negatively affect the applicability of models. Sometimes the results are significant. Sometimes they are extremely bad and we do not always understand why.
The variability in predictions: significantly different predictions may be obtained by using different conventional SRGMs on the same application data set. Differing predictive accuracies can also be obtained by applying the same model on different application data sets. Even more confusingly, different comparative techniques may indicate different predictive qualities of the same model with the same application data set.

The strong distributional assumptions about the times between failures: the incompatibility between the assumptions of the conventional models and a specific development/data collection scenario may cause inaccuracy in predictions.

The lack of enough information: many SRGMs use the test data as the only input. In view of the known lack of good data this does not seem like a formula for success. Even if care is taken in predicting the operational profile for a system under development, software test environments are very different from user environments. Exclusively using test data to predict reliability in use will not usually give accurate predictions.

The accuracy of the collected data: collection of data is extremely important, not only for measuring software reliability but also for improving it. Software reliability measures will only be as accurate as the data that is input into them. The less variance in the data the more accurate and usable the model will be.

The environment sensitivity: if the software system is being tested in a different environment, the failure history of the past will not reveal these changes, and good prediction of future behavior cannot be expected.

A number of approaches have been proposed to avoid these difficulties:

Using more than one model: always evaluate more than one model if there is insufficient prior experience to identify a single model that works effectively in a specific development context. Then do a comparative study to select the most appropriate model for the software under test. Examples of comparative techniques that can help in choosing the most appropriate model can be found in [Abdel-Ghaly et al. (1986) and Akaike (1974)].
- **Try to let the data help in choosing a model (or a group of models):** performing reliability trend analysis will help to choose a suitable model. In this criterion, the curve of the observed failure data is compared with the curve of the model. If the two curves match, then there are more opportunities of obtaining accurate prediction results by using that model. Several statistical tests have been presented for determining trends in observed data. These include the Kendall, Spearman and Laplace tests. The Laplace test is very useful in determining the reliability trends. By using this test four possible types of reliability trends can be obtained after the trend analysis and so a model (or a group of models) can be chosen accordingly e.g. [Lyu (1996), Kanoun and Laprie (1994), and Musa et al. (1987)].

- **Consider the assumptions of the models:** statistical models always encapsulate assumptions about their domain of application. It is not always easy to find explicit statements of all the underlying assumptions in a model. Nevertheless, selecting those models whose assumptions are most appropriate for the specific development and testing environment for the software being evaluated should lead to higher predictive accuracy. As an alternative, the use of non-parametric models may relax the assumptions that are made by the conventional parametric models and may give better prediction results, although usually with more stringent data requirements. For examples of these, see [Sofer and Miller (1991) and Barghout et al. (1997)].

- **Use some model refinement approaches:** a number of techniques have been proposed to improve the predicative accuracy of conventional SRGMs by reviewing their predicative power as they are applied to the data, and dynamically updating the models (examples of those techniques can be found in [Brocklehurst et al. (1990), Brocklehurst and Littlewood (1992), and Lyu and Nikora (1991)].

- **Add more information to the conventional models:** instead of using the failure times as the only input to the software reliability models, we may need to consider and incorporate some environmental factors during the software development process to refine the existing models and get more accurate reliability measurement. Several new software reliability models that incorporate environmental factors are discussed in [Pham (2000), Xia and Kumar (1992), and Huang (2005)].
Using more accurate tools for collecting the data: it is very important that what must be measured is determined before data collection begins. Data collection is expensive and will only have a payback if it is done effectively and efficiently. Determine initially what it is you want to know about your software process and then determine how to collect that data. The data must be collected in a consistent manner from that point on to reduce in the data variability that will make any software metric or model become less accurate or invalid.

Ultimately, the problems associated with the art of software reliability modeling are very complicated. But we are optimistic that this art will continue to develop. This will require developing a standard modeling approach, and a body of knowledge to interpret the phenomena that are being modeled. So reliability as a problem that has a critical relationship with quality will not simply disappear, but more studies and modeling challenges are still there to be addressed.

7.2 Suggestions for Future Research

So far, little attention has been paid to non-parametric models and their data-analysis approaches. Since there is so little agreement about fundamental modeling assumptions in this field, this may be a fruitful area for investigation.

It is worthy to incorporate a factor of learning processes during the testing phase and construct developed SRGMs according to this improvement. We believe that little is currently known in that direction. The best advice that can be given to potential users is to increase studies on optimal release policies based on different cost and developed prediction functions.

Maybe, it is good to consider if there are models that make weaker assumptions about the nature of the test process. Those models may be less precise in general, but could be more applicable to several test data sets.
Further studies are to attempt wider applications when the inter-failure times belong to some other unstudied distributions. The most important suggestion is to discuss the problems from the Bayesian framework.

The use of Bayesian networks is an approach that has been used to make a prediction of residual faults in a software product before final system testing is complete [Fenton et al. (2007)]. Also, the prior can have a dominant influence if there is a strong disagreement between the prior prediction and the test results. We may need to investigate if we could take a prediction of residual faults using a Bayesian network, and then use this as a prior belief in a Bayesian reliability growth model. Overall, our feeling is that we may have achieved the limits of what is possible using statistical analysis of testing data alone, and that an innovative solution involving a deeper analysis of the software product, and possibly the processes used to develop it, is needed in order to move software reliability assessment to a higher level of precision.
Appendix A.: Proofs of some relations

A.1 Proof (1)

From equation (4.18) with \( b(t) = b + (1 - r) \frac{H(t)}{a} \), we have

\[
\frac{dH(t)}{r + (1 - r) \frac{H(t)}{a} [a - H(t)]} = b dt.
\]

Integrating both sides with respect to \( t \) from \( t = 0 \) to \( t \), we get on using partial fractions:

\[
\int_0^t \frac{(1 - r) dH(t)}{ar + (1 - r) H(t)} + \int_0^t \frac{dH(t)}{a - H(t)} = \int_0^t b dt.
\]

Hence:

\[
\left[ \ln \left( ar + (1 - r) H(t) \right) \right]_0^t - \left[ \ln \left( a - H(t) \right) \right]_0^t = bt.
\]

Since \( y_0 = H(0) = 0 \), this gives

\[
\ln \left( ar + (1 - r) H(t) \right) - \ln \left( ar \right) - \ln \left( a - H(t) \right) + \ln a = bt.
\]

\[
\therefore \ln \left[ \frac{r \left( a - H(t) \right) + H(t)}{a - H(t)} \right] - \ln r = bt.
\]

\[
\therefore \ln \left[ \frac{1}{r} \left( r - 1 + \frac{a}{a - H(t)} \right) \right] = bt.
\]
\[ a - H(t) = \frac{a}{r e^{bt} + 1 - r}. \]

\[ H(t) = a \left( 1 - \frac{1}{r e^{bt} + 1 - r} \right) \]

\[ = a \left( \frac{1 - e^{-bt}}{1 + ce^{-bt}} \right), \quad a > 0, \ b > 0, \ c = \frac{1 - r}{r} > 0. \]

### A.2 Proof (2)

From equation (4.18) with \( a = a(1 - p e^{-\nu t}) \) and \( b(t) = b \), we have

\[ \frac{dH(t)}{a(1 - p e^{-\nu t}) - H(t)} = b dt, \]

or

\[ \frac{dH(t)}{dt} + bH(t) = ab(1 - p e^{-\nu t}) \]

which is a first order differential equation that can be transformed to an exact one by multiplying both sides by the integrating factor \( e^{bt} \), hence the solution of this equation can be obtained as follows:

\[ H(t) = e^{-bt} ab \int \left( e^{bt} - p e^{-(b-\nu)u} \right) du \]

\[ = ab e^{-bt} \left\{ \frac{e^{bt} + pe^{-(b-\nu)t}}{b} - \frac{1 - p}{b} \right\} \]

\[ = a \left\{ 1 - (v - b + bp)e^{-bt} - bpe^{-vt} \right\} \]

\[ = a \left\{ 1 - \frac{f - bs}{v - b} \right\}. \]
where \( f = (v - b + bp)e^{-bt} \), \( s = pe^{-vt} \) and \( v \neq b \).

In the case of \( v = b \), we have
\[
\frac{dH(t)}{dt} + bH(t) = ab(1 - pe^{-bt}).
\]

Multiplying both sides, by the integrating factor \( e^{bt} \) and integrating with respect to \( t \) we obtain
\[
H(t)e^{bt} = ab \int_0^t (e^{bt} - p)dt.
\]

Hence
\[
H(t) = abe^{-bt}\left(\frac{e^{bt}}{b} - pt - \frac{1}{b}\right) = a\left(1 - (bp + 1)e^{-bt}\right).
\]
Appendix B: References


Krause, P.J., Freimut, B. and Suryn, W. (2003). New Directions in Measurement for Software Quality Control, this work was supported in part by the European Community funded R&D Project MODIST [IST-2000-28749].


