A Study on Markov Process Modeling for Software Reliability Assessment

Dissertation submitted in partial fulfillment for the
degree of Doctor of Informatics and Data Science

Siqiao Li

Under the supervision of
Professor Tadashi Dohi

Dependable Systems Laboratory,
Informatics and Data Science Program
Graduate School of Advanced Science and Engineering,
Hiroshima University, Higashi-Hiroshima, Japan

March 2023

Abstract

In the typical waterfall development model, the software development process consists of 5 steps: (i) requirement/specification analysis, (ii) preliminary and detailed design, (iii) coding, (iv) testing/verification, and (v) maintenance. In the testing phase especially, software faults are detected and removed as much as possible to meet high software reliability requirements. In other words, the success of software testing leads to guaranteeing the quality of software. Since software reliability is considered as one of the most fundamental and significant attributes of software quality, considerable attention has been paid to improving software testing. At the same time, since software testing is quite expensive, the quantification of software reliability is also another important issue in the verification phase. The quantitative software reliability is generally defined as the probability that software failures caused by faults do not occur in a given time interval after the release. It is common to describe the probabilistic behavior of the fault-detection process in testing phases by any stochastic counting process. The software reliability defined in the above cannot be measured directly in the field, so that stochastic models, which are called software reliability models (SRMs), can be utilized to assess the quantitative software reliability. In fact, a great number of SRMs have been developed to control/monitor software testing processes as well as to evaluate the quantitative software reliability during the last four decades.

In this thesis, we propose numerous novel SRMs, based on the homogeneous Markov processes (HMPs) and non-homogeneous Markov processes (NHMPs). We formulate the maximum likelihood (ML) estimation of our SRMs and perform the software reliability analysis with the fault count time-interval data (group data), fault count time-domain data, and time-dependent software metrics data, which can be observed in the software industry. By comparing our SRMs with the representative existing SRMs, we evaluate the performances of models comprehensively. In Chapter 1, we introduce the definition of HMP and NHMP-based software reliability modeling, including the well-known non-homogeneous Poisson process (NHPP)-based modeling framework. In Chapter 2, we focus on the pure birth process (HMP) to describe software fault counts, called geometric de-eutrophication SRM. We provide some useful results to han-

dle the software fault count group data. Two types of SRMs are considered; Moranda SRM (1975) and Gaudoin-Soler SRM (1992), where the former is a modification of the well-known Jelinski-Moranda SRM (1972), having a software fault detection rate with geometrically decreasing reduction, the latter is an extension of Moranda SRM having another software fault detection rate with exponential decay. Chapter 3 primarily focuses on the finite-failure (type-I) NHPP-based SRMs and infinite-failure (type-II) NHPP-based SRMs. For describing the software fault-detection time distribution, we postulate 29 representative probability distribution functions that can be categorized into the generalized exponential distribution family, the extreme-value distribution family, the Burr-type distribution family, and the Lindley-type distribution family. We verify the usefulness of our type-I and type-II NHPP-based SRMs and confirm how well they make decisions in software reliability assessment, We compare the goodness-of-fit and predictive performances with the representative existing NHPP-based SRMs. In Chapter 4, we propose local polynomial SRMs, which can be categorized into a semi-parametric modeling framework. Our models belong to the common NHPP-based SRMs but possess a flexible structure to approximate an arbitrary mean value function by controlling the polynomial degree. More specifically, we develop two types of local polynomial NHPP-based SRMs; finite-failure and infinite-failure SRMs, which are substantial extensions of the existing NHPP-based SRMs in a similar category. Chapter 5 discusses the so-called proportional intensity-based software reliability models (PI-SRMs), which are extensions of the common NHPP-based SRMs, and describe the probabilistic behavior of the software fault-detection process by incorporating the time-dependent software metrics data observed in the development process. In Chapter 6, we focus on NHMPs, which are generalizations of the well-known HMPs and NHPPs, and compare two SRMs that can be classified into a generalized binomial processes (GBPs) and generalized Polya processes (GPPs). GBP and GPP are also characterized respectively as a Markov inverse death process and a Markov birth process, with state- and time-dependent transition rates, respectively. We develop a unified software reliability modeling framework based on the NHMPs and apply them to the software reliability prediction. Throughout numerical examples with the fault count data observed in

actual closed-source software (CSS) and open-source software (OSS) development projects, we compare two NHMP-based SRMs (GBP and GPP) in terms of the goodness-of-fit and predictive performances, in addition to the quantitative software reliability assessment. We also consider software release problems with these generalized SRMs and, investigate the impact on the software release decision. Finally, some conclusions and remarks are given in Chapter 7.

Acknowledgements

First and foremost, I would like to extend my sincere gratitude to Professor Tadashi Dohi, the supervisor of my study, for his valuable guidance and kind advice in every stage of the writing of this thesis. I could not have completed my thesis without his enlightening instruction and impressive patience.

Also, my thanks go to Professor Hiroyuki Okamura, Professor Shaoying Liu, Associate Professor Tadashi Shima, and Professor Yasuhiko Morimoto for their invaluable comments, useful suggestions, and warm encouragement. I also wish to thank the JST SPRING (Next-Generation Fellowship) for financial support in my Ph.D study.

Finally, it is my special pleasure to acknowledge the hospitality and encouragement of the past and present members of the Dependable Systems Laboratory, Hiroshima University. Last but not the least, I would like to thank my parents for supporting me spiritually throughout writing this thesis and my life in general.

Contents

List of Abbreviations

AIC Akaike information criterion

CSS Closed-source software

CTMC Continuous-time Markov chain

GBP Generalized binomial process

GPP Generalized Polya process

HMP Homogeneous Markov process

HPP Homogeneous Poisson process

IHR increasing hazard rate

LLF Log likelihood function

ML Maximum likelihood

MSE Mean squared error

MTBF Mean time between failures

NHMP Non-homogeneous Markov process

NHPP Non-homogeneous Poisson process

OSS Open-source software

PHM Proportional hazard model

PI-SRM Proportional intensity-based software reliability model

PLL Predictive Log Likelihood

PMSE Predictive mean squared error

R.V. Random variable

SRATS Software reliability assessment tool on the spreadsheet

SRM Software reliability model

c.d.f. Cumulative distribution function

i.i.d. independent and identically distributed

p.d.f. Probability density function

p.m.f. Probability mass function

List of Symbols

$a, b, c, d, \omega, \epsilon, \zeta, \mu_i$	constant parameters in SRMs
$d(t)$	debug rate function
$F(t)$	cumulative distribution function
$F_n(x)$	cumulative distribution function of the inter-failure time X_n
$g(\boldsymbol{x})$	covariate function of PI-SRMs
$G_n(t)$	cumulative distribution function of the n-th failure times
m	number of software fault count data
$M(t)$	expect number of detected faults at time t
N	total number of inherent faults in a software program
$N(t)$	cumulative number of software fault counts by time t
$P_n(t)$	steady-state transition probability in state n at time t
$Q_n\,(t)$	conditional probability function
t_i	i-th $(i = 1, 2, \cdots, m)$ fault detection time data
t_e	observation (truncation) point of fault detection time data
T_n	n-th software fault detection time
X_n	time interval between the n-th and $(n+1)$-st software faults
$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{\nu}$	parameter vector in SRMs
$\Lambda(t)$	cumulative (baseline) intensity function
λ_n	state-dependent transition rate (intensity function)
$\lambda(t)$	time-dependent transition rate (intensity function)
$\lambda_n(t)$	state- and time-dependent transition rate (intensity function)
$\kappa(t)$	baseline intensity function
$\mathcal{L}(\cdot)$	likelihood function
$\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\nu}}$	maximum likelihood estimates of parameters $\boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\nu}$
τ_i	i-th $(i = 1, 2, \cdots, m)$ software testing time

$\Gamma(\cdot)$	standard gamma function
$\mathrm{erfc}(\cdot)$	complementary error function
$\ln(\cdot)$	natural logarithmic function
$\{n_i; i = 1, 2, \cdots, m\}$	cumulative number of software fault counts by time τ_i
$\{t_i, i = 1, \ldots, m\}$	i-th $(i = 1, 2, \cdots, m)$ time-domain data
$\{(\tau_i, n_i), i = 1, 2, \ldots, m\}$	i-th $(i = 1, 2, \cdots, m)$ time-interval data (group data)
$\lceil x \rceil$	ceiling function which is the largest integer less than x

Chapter 1

Introduction

1.1 Summary of Existing Software Reliability Modeling Frameworks

Throughout software development processes, software reliability models (SRMs) have been extensively used in the verification and validation phase to quantify the software reliability, which is defined as the probability that software faults are not detected in the remaining testing period or that software failure caused by software faults do not occur in the operational phase after the release to the user or market. During the almost last five decades, a great number of SRMs have been developed by many authors [1, 2, 3]. Especially, the homogeneous Markov process (HMP)-based SRMs and non-homogeneous Poisson process (NHPP)-based SRMs have gained much popularity for describing the stochastic behavior of the cumulative number of software faults detected in the testing phase, because of their tractability and goodness-of-fit performance.

The majority of SRMs developed in the past is *estimable* and, possesses the so-called Markov property. For instance, the most classical SRMs by Jelinski and Moranda [4], Moranda [5], Xie [6] are categorized into HMPs with different state-dependent transition rates (equivalently, pure Markov inverse death process and pure Markov birth process). The NHPP-based SRMs [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20] are also Markov processes with different time-dependent transition rates.

1.1.1 HMP-based Software Reliability Modeling

Let $\{N(t), t \geq 0\}$ denote the cumulative number of software faults detected up to the system testing time t. Since the software fault counting process $N(t)$ can be considered as a non-decreasing stochastic point process taking non-negative integer values, it is useful to describe the dynamic behavior by using a Markov counting process. Let

$$P_n(t) = \Pr\{N(t) = n | N(0) = 0\}, \ n = 0, 1, 2, \cdots \tag{1.1}$$

be the steady-state transition probability. The HMP is described by the state-dependent transition rate λ_n $(n = 0, 1, 2, \cdots)$. Suppose that there exist N software faults remaining in a software before the system testing, and that the software fault count process is given by an inverse birth process (see Fig. 1.1 (a)) with an absorbing state $n = N$. Jelinski and Moranda [4] considered this type of HMP with termination and assumed the transition rate $\lambda_n = (N - n)b$ $(n = 0, 1, \ldots, N - 1)$, where N is the residual number of software faults before the testing (non-negative integer) and b is the constant fault-detection rate when each of software fault-detection times in N population is independent and identically distributed exponential random variable. Since the Kolmogorov forward equations are given by

$$\frac{\mathrm{d}}{\mathrm{d}t} P_0(t) = -\lambda_0 P_0(t), \tag{1.2}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} P_n(t) = \lambda_{n-1} P_{n-1}(t) - \lambda_n P_n(t), \ n = 1, 2, \cdots, N-1, \tag{1.3}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} P_N(t) = \lambda_{N-1} P_{N-1}(t) \tag{1.4}$$

with the boundary conditions $P_0(0) = 1$ and $P_n(0) = 0$ $(n = 1, \ldots, N)$, it is straightforward to obtain the probability mass function (p.m.f.) [107]:

$$P_n(t) = \binom{N}{n} \{1 - e^{-bt}\}^n e^{-b(N-n)t}, \quad n = 0, 1, \cdots, N, \tag{1.5}$$

which is a binomial p.m.f. Hence the process $N(t)$ terminates at $n = N$ with probability one.

Moranda [5] assumed another transition rate $\lambda_n = ab^n$ under the assumption of $N \to \infty$, where a and b are two model parameters with apparently no physical interpretation, and proposed the so-called the geometric de-eutrophication type

SRM to describe the software fault-count process. Since it is a pure birth process (see Fig. 1.1 (b)), the Kolmogorov forward equations are given by

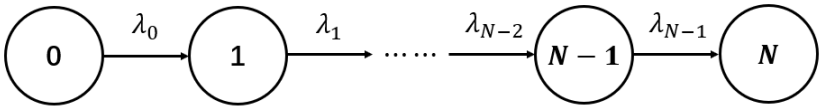
$$\frac{\mathrm{d}}{\mathrm{d}t}P_0(t) = -\lambda_0 P_0(t), \tag{1.6}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_n(t) = \lambda_{n-1}P_{n-1}(t) - \lambda_n P_n(t),\ n = 1, 2, \cdots \tag{1.7}$$

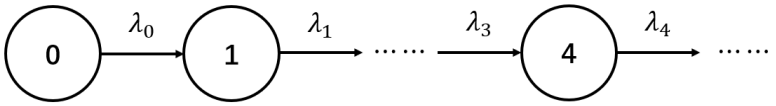
with the boundary conditions $P_0(0) = 1$ and $P_n(0) = 0$. From the well-known nature of the pure birth process type of HMP, it holds (see *e.g.*, [21]) that the p.m.f. is a unique solution of

$$P_n(t) = \lambda_{n-1}e^{-\lambda_n t}\int_0^t e^{\lambda_n x}P_{n-1}(x)dx, \tag{1.8}$$

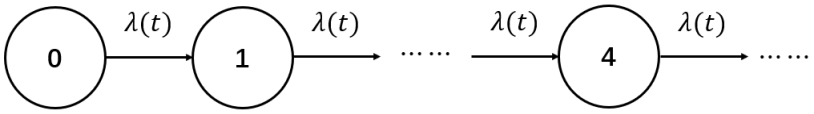
if $\sum_{n=0}^{\infty}\lambda_n^{-1} = \infty$. Boland and Singh [22] obtained a closed form of the c.d.f. with $\lambda_n = ab^n$ for Moranda's geometric de-eutrophication type SRM. Fig. 1.2 (a) and (b) show the schematic illustrations of the transition rates in Jelinski and Moranda SRM [4] and Moranda SRM [5], respectively.



(a) HMP with termination.



(b) HMP without termination.



(c) NHPP.

Figure 1.1: Transition diagrams.

1.1.2 NHPP-based Software Reliability Modeling

As a well-known Markov process, NHPP is regarded as an alternative to the classical homogeneous Poisson process (HPP). If the intensity at time point t in the definition of HPP is given by a function $\lambda(t)$ with respect to t, then an HPP can be described by an NHPP. More specifically, if a stochastic counting process $\{N(t), t \geq 0\}$ is non-negative and non-decreasing, it becomes an NHPP under the following assumptions.

- NHPP has independent increments, so the number of occurrences in a specific time interval depends on only the current time t and does not on the past history of the process, which is also known as the Markov property.
- Initial state of the process is given by $N(0) = 0$.
- The occurrence probability of one event in a given time period $[t, t + \Delta t)$ for an NHPP is defined by $\Pr\{N(t + \Delta t) - N(t) = 1\} = o(\Delta t) + \lambda(t)\Delta t$. The function $\lambda(t)$ is an absolutely continuous function, which is named the *intensity function* of NHPP, and Δt is recognized as an infinitesimal period of time.
- NHPP has negligible probability for two or more events occurring in $[t, t + \Delta t)$, i.e., $\Pr\{N(t + \Delta t) - N(t) \geq 2\} = o(\Delta t)$, where $\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$ and $o(\Delta t)$ is the higher-order term of Δt.

For an arbitrary non-negative and absolutely continuous function of time, $\lambda(t)$, consider the Kolmogorov forward equations:

$$\frac{\mathrm{d}}{\mathrm{d}t} P_0(t) = -\lambda(t) P_0(t), \tag{1.9}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} P_n(t) = \lambda(t) P_{n-1}(t) - \lambda(t) P_n(t), \; n = 1, 2, \cdots . \tag{1.10}$$

By solving the above difference-differential equations with the initial conditions $P_0(0) = 1$ and $P_n(0) = 0$ $(n = 1, 2, \ldots)$, it is immediate to derive

$$P_n(t) = \frac{M(t)^n}{n!} e^{-M(t)}, \quad n = 0, 1, 2, \cdots , \tag{1.11}$$

so that $N(t)$ is a non-homogeneous Poisson process (NHPP) with the mean value function:

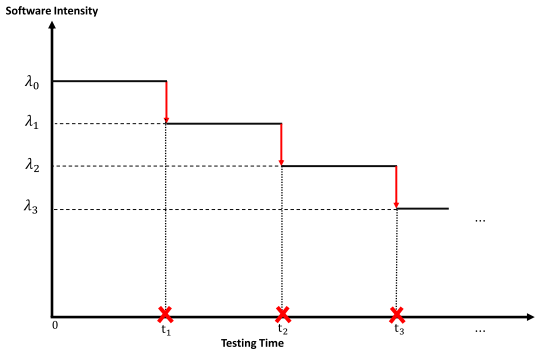
$$\mathrm{E}[N(t)] = M(t) = \int_0^t \lambda(x) dx. \tag{1.12}$$

From the Poisson nature, it holds that $\mathrm{E}[N(t)] = \mathrm{Var}[N(t)]$. This unusual feature without apparent empirical interpretation is called *the equity-dispersion*, so the expected cumulative number of software faults must be exactly same as its variance in this modeling assumption. The time-dependent transition rate $\lambda(t)$ in Equations (1.9) and (1.10) is called *the intensity function* in the NHPP and is independent of the state n. In Fig. 1.1 (c), we depict the transition diagram of the NHPP with time-dependent transition rate $\lambda(t)$. If the function $\lambda(t)$ is decreasing (increasing) in t, then the software tends to be reliable (unreliable) as the testing time goes on (see Fig.1.2 (c) and (d)). This model does not focus on the microscopic behavior of each software fault count, but describes the time-dependent macroscopic trend in the software fault intensity. By modeling the software failure time, Kuo and Yang [23] classified NHPP-based SRMs into general order statistics SRMs and record value statistics SRMs. The same authors [23] referred an alternative and simpler classification by dividing NHPP-based SRMs into two types; finite-failure (type-I) and infinite-failure (type-II) NHPP-based SRMs with the mean value functions, which are defined as the expected cumulative number of software failures.

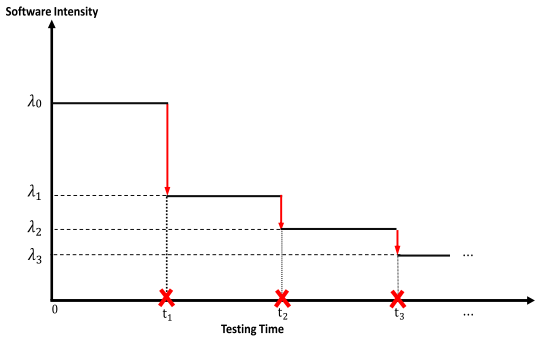
1.1.3 Maximum Likelihood Estimation

Once the intensity function (transition rate) is determined in HMP- and NHPP-based SRMs, the commonly used technique to estimate the model parameters is the maximum likelihood estimation by maximizing the log likelihood function (LLF). In general, there are two types of software fault count data; time data and group data. The time data can be also called the fault count time-domain data. For $t_0 = 0$, we observe m fault detection times, t_i $(i = 1, 2, \cdots, m)$, where t_e $(\geq t_m)$ denotes the observation (censoring) point of time. For the time-domain data $(t_1, t_2, \cdots, t_m; t_e)$, the likelihood functions of the HMP- and NHPP-based SRMs with the time truncation are given by

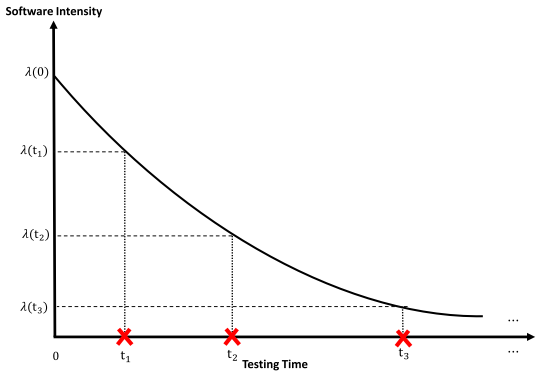
$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \prod_{i=1}^{m} \lambda_i(\boldsymbol{\theta}) e^{-\lambda_i(\boldsymbol{\theta})(t_i - t_{i-1})} e^{-\lambda_m(\boldsymbol{\theta})(t_e - t_m)} \\ &= \exp[-\lambda_m(\boldsymbol{\theta}) t_e] \prod_{i=1}^{m} \lambda_i(\boldsymbol{\theta}), \end{aligned} \tag{1.13}$$



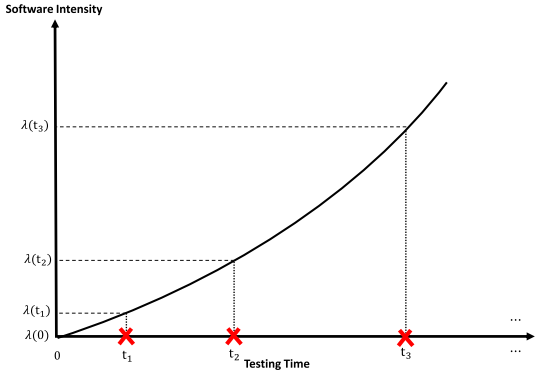
(a) Jelinski-Moranda SRM (HMP): $\lambda_n = (N - n + 1)b$ with a constant downward jump.



(b) Moranda SRM (HMP): $\lambda_n = ab^n$ $(0 < b < 1)$ with a decreasing upward jump.



(c) NHPP-based SRM with decreasing intensity function.



(d) NHPP-based SRM with increasing intensity function.

Figure 1.2: Representative baseline models in SRMs.

and

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \prod_{i=1}^{m} \lambda(t_i;\boldsymbol{\theta}) e^{-\int_{t_{i-1}}^{t_i} \lambda(x;\boldsymbol{\theta})dx} e^{-\int_{t_m}^{t_e} \lambda(x;\boldsymbol{\theta})dx} \\ &= \exp[-M(t_e;\boldsymbol{\theta})] \prod_{i=1}^{m} \lambda(t_i;\boldsymbol{\theta}), \end{aligned} \tag{1.14}$$

respectively, where $\boldsymbol{\theta}$ is the model parameter vector. In the failure truncation, say $t_e = t_m$, the corresponding likelihood functions are given by replacing t_e by t_m in Equations (1.13) and (1.14). Taking the logarithm of Equations (1.13) and (1.14), we obtain the log likelihood functions:

$$\ln \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \ln \lambda_i(t_i;\boldsymbol{\theta}) - \lambda_m(t_e;\boldsymbol{\theta}), \tag{1.15}$$

$$\ln \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \ln \lambda(t_i;\boldsymbol{\theta}) - M(t_e;\boldsymbol{\theta}). \tag{1.16}$$

By maximizing the log likelihood function $\ln \mathcal{L}(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, we obtain the maximum likelihood estimates $\hat{\boldsymbol{\theta}}$.

When the group data (τ_i, n_i) $(i = 0, 1, \cdots, m)$ with $(\tau_0, n_0) = (0, 0)$ are available, for HMP-based SRMs, the likelihood function is represented by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \Pr\Big\{N(\tau_1) = n_1, N(\tau_2) = n_2, \cdots, N(\tau_m) = n_m\Big\} \\ &= \prod_{i=1}^{m} \Pr\{N(\tau_i) = n_i \mid N(\tau_{i-1}) = n_{i-1}\}, \end{aligned} \tag{1.17}$$

from the Markov property. So, if the conditional transition probability $\Pr\{N(\tau_i) = n_i \mid N(\tau_{i-1}) = n_{i-1}\}$ $(i = 1, 2, \cdots, m)$ is available for HMP-based SRMs, the corresponding likelihood function is obtained explicitly. For instance, the likelihood function for Jelinski and Moranda SRM [4] is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{m} \binom{N - n_{i-1}}{n_i - n_{i-1}} \{1 - e^{-b\tau_i}\}^{n_i - n_{i-1}} e^{-b(N-n_i)\tau_i} \tag{1.18}$$

for $\boldsymbol{\theta} = (N, b)$, but the likelihood functions for Moranda SRM [5] and Xie SRM [6] have to be calculated algorithmically.

The likelihood function for the unknown parameters $\boldsymbol{\theta}$ for NHPP-based SRMs is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{m} \left[\frac{[M(\tau_i;\boldsymbol{\theta}) - M(\tau_{i-1};\boldsymbol{\theta})]^{n_i - n_{i-1}}}{(n_i - n_{i-1})!}\right] \\ \times e^{-[M(\tau_i;\boldsymbol{\theta}) - M(\tau_{i-1};\boldsymbol{\theta})]}, \end{aligned} \tag{1.19}$$

so that the log likelihood function is represented as

$$\ln \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \Big\{ (n_i - n_{i-1}) \ln[M(\tau_i;\boldsymbol{\theta}) - M(\tau_{i-1};\boldsymbol{\theta})] - \ln[(n_i - n_{i-1})!] \Big\} - M(\tau_m;\boldsymbol{\theta}). \tag{1.20}$$

In the above way, the maximum likelihood estimation for NHPP-based SRMs is trivial even for both time and group data. However, it is not always easy to handle the group data for HMP-based SRMs if the analytical forms of transition rates are not available.

1.2 Software Fault Count Data

Most observable software testing data in the industry are fault count data, as it is common to test/debug software in the distributed testing environment. Generally, it is possible to observe software fault count data in two categories; software fault count time-domain data and software fault count time-interval data (group data). In this thesis, we employ thirteen time-domain data sets for closed-source software (CSS) systems and twelve group data sets for eight CSS and four open-source software (OSS) systems on the software fault count. A set of fault detection times measured with CPU time is called the fault count time-domain data. Suppose that m software faults are detected, where the time sequence is given by $\boldsymbol{D} = \{t_1, t_2, \ldots, t_m\}$. On the other hand, a group data $\boldsymbol{I} = \{(\tau_i, n_i), i = 1, 2, \ldots, m\}$ consists of the number of faults detected in fixed time intervals measured with the calendar time, $(\tau_{i-1}, \tau_i]$ $(i = 1, 2, \ldots, m)$. Each record of the group data (τ_i, n_i) is given by a pair of the observation time τ_i and the cumulative number of software faults detected by time τ_i. In this thesis, we list these data in order from DS1 to DS25. We show the details of these data sets in Table 1.1. In Chapter 5, we also consider an extension of the common NHPP-based SRM and describe the probabilistic behavior of the software fault detection process by incorporating software time-dependent metric data. In Table 1.1 (iv), we show the four software time-dependent metric data we used, DS26∼DS29. It is not difficult to find that DS26 and DS8, DS27 and DS14, DS28 and DS15, and DS29 and DS5 come from the same source, respectively.

Table 1.1: Software Fault Count Data Sets.

(i) Time-domain data (CSS development projects).

Data set	No. faults	Testing length (CPU time)	Source	Nature of system
DS1	54	108708	SYS2 [24]	Real time command and control system
DS2	38	233700	S10 [24]	Real time command and control system
DS3	38	67362	SYS3 [24]	Real time command and control system
DS4	41	4312598	S27 [24]	Military application
DS5	53	52422	SYS4 [24]	Real time command and control system
DS6	73	5090	Project J5 [1]	Real time command and control system
DS7	101	19572126	S17 [24]	Real time command and control system
DS8	136	88682	SYS1 [24]	Real time command and control system
DS9	24	1095.88	S14C [24]	Real time commercial subsystem
DS10	129	89040	SRC2 [1]	Single-user workstation
DS11	197	50236822	SS4 [24]	Operating system
DS12	104	15369.5	SRC3 [1]	Single-user workstation
DS13	397	108890	SRC1 [1]	Single-user workstation

(ii) Group data (CSS development projects).

Data set	Testing weeks	No. faults	Source	Nature of system
DS14	17	54	SYS2 [24]	Real time command and control system
DS15	14	38	SYS3 [24]	Real time command and control system
DS16	19	120	Release2 [25]	Tandem software system
DS17	12	61	Release3 [25]	Tandem software system
DS18	14	9	NASA -supported project [26]	Inertial navigating system
DS19	20	66	DS1 [27]	Embedded application for printer
DS20	33	58	DS2 [27]	Embedded application for printer
DS21	30	52	DS3 [27]	Embedded application for printer

(iii) Group data (OSS development projects).

Data set	Operating months	No. faults	Source	Project
DS22	121	379	[28]	Video game emulation for macOS
DS23	107	381	[29]	JavaScript framework for building web interfaces
DS24	62	260	[30]	Screenshot software for Windows
DS25	96	367	[31]	Math typesetting for the web

(iv) Time-dependent metric data (CSS development projects).

Data set	No. Faults	Testing weeks	Source	Nature of system
DS26	136	21	SYS1 [24]	Real time command and control system
DS27	54	17	SYS2 [24]	Real time command and control system
DS28	38	14	SYS3 [24]	Real time command and control system
DS29	53	16	SYS4 [24]	Real time command and control system
Metrics Data:	Failure identification work, Execution time, Computer time-failure identification.			

Chapter 2

HMP-based Software Reliability Models (SRMs) with group data

In this chapter, we focus on a pure birth process to describe software fault count, called the geometric de-eutrophication SRM, and provide some useful results to handle the software fault count group data. Two types of SRMs are considered; Moranda SRM [32, 33], (M-SRM) and Gaudoin-Soler SRM [34] (G & S-SRM), where the former is a modification of the well-known J&M-SRM [4] having a software fault detection rate with geometrically decreasing reduction, the latter is an extension of M-SRM [32, 33] having another software fault detection rate with exponential decay. First, we note that these two SRMs; M-SRM and G&S-SRM, are essentially identical. Unfortunately, it is emphasized that the group data analysis with the geometric de-eutrophication SRM has not been done yet in the literature, so M-SRM [32, 33] and G&S-SRM [34] handled only the time domain data. Boland and Singh [35] and Vasanthi and Arulmozhi [36] gave the fundamental results to characterize M-SRM[32, 33], but did not apply their results to the maximum likelihood estimation with the group data. In other words, the geometric de-eutrophication SRM has not been fully proven whether it could accurately describe the software fault detection behavior in the testing phase of the actual software development project. This fact is really surprising because M-SRM [32, 33] is one of the most classical SRMs and has not been investigated in the viewpoint of quantification of software reliability in the plausible group data circumstance for several decades.

2.1 Preliminary

Jelinski and Moranda [4] proposed the earliest SRM, which is called J&M-SRM, and described the software fault count process as a homogeneous Markov death process, where the software fault-detection rate is proportional to the remaining number of faults in a software program. They dealt with the time-domain data and estimated the model parameters by means of the maximum likelihood method. Shanthikumar [38] and Xie [39] showed that J&M-SRM is essentially the same as a binomial process in terms of stochastic counting processes and estimated the model parameters with the group data. Miller [40] also showed that J&M-SRM could be derived from exponential order statistics. In this way, the most well-known J&M-SRM can be handled with both time-domain and group data. Another representative SRMs are the NHPP-based SRMs, which can also deal with both time domain data and group data (see Musa, Iannino and, Okumoto [2]).

In this chapter, we focus on a pure birth process to describe software fault count, called the geometric de-eutrophication SRM, and provide some useful results to handle the software fault count group data. Two types of SRMs are considered; Moranda SRM [32, 33] (M-SRM) and Gaudoin-Soler SRM [34] (G&S-SRM), where the former is a modification of the well-known J&M-SRM [4] having a software fault detection rate with geometrically decreasing reduction, the latter is an extension of M-SRM [32, 33] having another software fault detection rate with exponential decay. First, we note that these two SRMs; M-SRM and G&S-SRM, are essentially identical. Unfortunately, it is emphasized that the group data analysis with the geometric de-eutrophication SRM has not been done yet in the literature, so M-SRM [32, 33] and G&S-SRM [34] handled only the time domain data. Boland and Singh [35] and Vasanthi and Arulmozhi [36] gave the fundamental results to characterize M-SRM [32, 33], but did not apply their results to the maximum likelihood estimation with the group data. In other words, the geometric de-eutrophication SRM has not been fully proven whether could accurately describe the software fault detection behavior in the testing phase of the actual software development project. This fact is really surprising because M-SRM [32, 33] is one of the most classical SRMs and has not been investigated in the viewpoint of quantification of software reliability

in the plausible group data circumstance for several decades.

2.2 De-eutrophication SRMs

In de-eutrophication SRMs, we suppose the non-increasing state-based fault detection (transition) rate in monotone time-homogeneous Markov processes such as pure birth process and pure death process, where the underlying assumption is that all the software faults detected are perfectly corrected, and no new faults are created through the system testing phase. In [37], the authors call this kind of model *de-eutrophication* SRM because the behaviour of removing software faults from a software program is very similar to the behaviour of cleaning pollutants from an enclosed body of water. In this sense, the well-known J&M-SRM [4] is also recognized as the earliest de-eutrophication SRM. More specifically, let N be the total number of inherent faults in a software program before the software testing. Then the failure rate in J&M-SRM, which is interpreted as the fault detection rate between the n-th and $(n+1)$-st software faults, is given by

$$\lambda_n = b(N-n),\ n = 0,\ 1,\ 2, \ldots, N-1, \tag{2.1}$$

where $b\ (>0)$ means a constant amount of contribution for software fault detection rate. Hence, the inter-fault-detection time intervals, X_n, are described by statistically independent exponential random variables with mean $1/\lambda_n\ (>0)$. Let $N(t)$ be the cumulative number of software faults detected by time $t\ (\geq 0)$. Then the probability mass function for J&M-SRM is given by [38, 39];

$$P_n(t) = \Pr\{N(t) = n \mid N(0) = 0\} = \binom{N}{n}\{1-e^{-bt}\}^n e^{-b(N-n)t},\ \ i = 0, 1, \cdots, N, \tag{2.2}$$

which is an elementary binomial distribution. Hence, the mean value function of $N(t)$ is given by $\mathrm{E}[N(t)] = N\{1 - e^{-bt}\}$.

On the other hand, M-SRM [32] assumes that the software faults detected in the early stage of software testing may be more serious than the others, and these faults may cause software failures that occurred in the beginning of testing. With the passage of time, it is assumed that the software fault-detection rate decreases geometrically. Based on these assumptions, the failure rate in M-SRM

is given by

$$\lambda_n = bk^n,\ n = 0,\ 1,\ 2, \ldots, \tag{2.3}$$

where b (> 0) and $k \in (0, 1)$ are constants. It is obvious that λ_n decreases monotonically but does not terminate at $n = N - 1$ dissimilar to J&M-SRM. This implies implicitly that an infinite number of software faults are contained in a software program. The main reason why M-SRM is called *geometric de-eutrophication* SRM is that λ_n in Equation (2.3) decreases geometrically as n increases. For the general pure birth process with transition rate λ_n ($n = 0, 1, 2, \ldots$), let $P_n(t) = \Pr\{N(t) = n \mid N(0) = 0\}$ denote the probability mass function or equivalently the steady-state transition probability. Then it holds that $\sum_{n=0}^{\infty} P_n(t) = 1$ if and only if $\sum_{n=0}^{\infty}(1/\lambda_n) = \infty$, so it turns out that M-SRM satisfies $\sum_{n=0}^{\infty} P_n(t) = 1$, and does not cause an explosion over a finite time t, say, $\sum_{n=0}^{\infty} P_n(t) < 1$. Let T_n denote the n-th software fault detection time with $T_n = \sum_{l=1}^{n} X_l$, where X_n is the time interval between the n-th and $(n + 1)$-st software faults. Then, the sequence $X_n,\ n = 0,\ 1, \ldots$, constitutes a geometric process [41]. From the exponential assumption, it is easy to derive the probability density function of X_n as $f_{X_n}(t) = bk^n \exp(-bk^n t)$ with t (≥ 0). Figure 1.1 (b) shows a schematic illustration of the fault-detection rate in M-SRM [32].

Gaudoin and Solar [34] considered another pure birth process with the failure rate;

$$\lambda_n = b \exp(-n \cdot a),\ n = 0,\ 1,\ 2, \ldots, \tag{2.4}$$

where a (> 0) and b (> 0) both specify the quality of the software debugging. If $a = 0$ in Equation (2.4), then the software fault detection rate becomes constant. If it is greater than 0, then the software fault detection rate decreases with time, and the reliability of software increases as more software faults are detected and corrected. In a fashion similar to M-SRM [32] the probability density function is given by $f_{X_n}(t) = b \exp(-na) \exp(-b \exp(-na)\, t)$. However, looking at Equations.(2.3) and (2.4), it can be seen that G&S-SRM is equivalent to M-SRM when $k = \exp(-a)$. Hereafter, we refer to M/G&S-SRM for the geometric de-eutrophication SRM. Next, we are interested in the cumulative number of software faults detected up to t (≥ 0) for M/G&S-SRM. Let $Q_n(t) = \Pr\{N(s + t) - N(s) = n | N(s) = m\}$ be the conditional probability

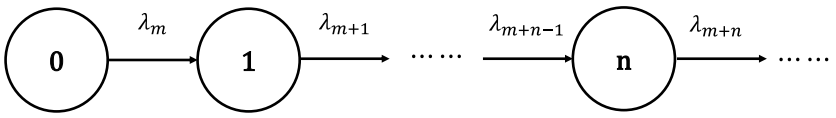


Figure 2.1: Transition diagram of the pure birth process without termination.

during the time interval $[s, s+t)$. The Kolmogorov forward equations are given by [35, 36]:

$$\frac{\mathrm{d}}{\mathrm{d}t}Q_0\left(t\right) = -\lambda_m Q_0\left(t\right), \tag{2.5}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}Q_n\left(t\right) = -\lambda_{m+n}Q_n\left(t\right) + \lambda_{m+n-1}Q_{n-1}\left(t\right), n = 1,\ 2, \ldots. \tag{2.6}$$

In Figure 2.1, we depict the transition diagram of a pure birth process with transition rate $\lambda_n(n = m, m+1, \ldots)$. Then the general solution in Equations (2.5) and (2.6) can be expressed by

$$Q_n(t) = \prod_{j=0}^{n-1}\lambda_{m+j}\sum_{j=0}^{n}\frac{\exp(-\lambda_{m+j}t)}{\prod_{l=0,l\neq j}^{n}(\lambda_{m+l}-\lambda_{m+j})}. \tag{2.7}$$

It is not so difficult to get the above result because the general solution of the ordinary linear differential equation can be provided. For the linear differential equation in Equation (2.6), the solution of $Q_n(t)$ satisfies the following recursive formula;

$$Q_n(t) = \exp(-\lambda_{m+n}t)\int_0^t \lambda_{m+n-1}Q_{n-1}(u)\exp(\lambda_{m+n}u - \lambda_m u)\mathrm{d}u. \tag{2.8}$$

From the initial condition $Q_0(t)$ in Equation (2.5) with $n = 0$, we have

$$Q_0\left(t\right) = \exp(-\lambda_m t). \tag{2.9}$$

Next, when $n = 1$, from Equation (2.8), we get

$$\begin{aligned} Q_1(t) &= \exp(-\lambda_{m+1}t)\int_0^t \lambda_m Q_0(u)\exp(\lambda_{m+1}u)\mathrm{d}u \\ &= \exp(-\lambda_{m+1}t)\int_0^t \lambda_m \exp(-\lambda_m u)\exp(\lambda_{m+1}u)\mathrm{d}u \\ &= \exp(-\lambda_{m+1}t)\frac{\lambda_m}{\lambda_{m+1}-\lambda_m}\left(\exp((\lambda_{m+1}-\lambda_m)t)-1\right) \\ &= \lambda_m\left(\frac{\exp(-\lambda_m t)}{\lambda_{m+1}-\lambda_m} + \frac{\exp(-\lambda_{m+1}t)}{\lambda_m - \lambda_{m+1}}\right). \end{aligned} \tag{2.10}$$

For $n = 2$, we obtain

$$\begin{aligned}
Q_2(t) &= \exp(-\lambda_{m+2}t)\int_0^t \lambda_{m+1}Q_1(u)\exp(\lambda_{m+2}u)\mathrm{d}u \\
&= \exp(-\lambda_{m+2}t)\lambda_{m+1}\lambda_m\left\{\left(\frac{1}{\lambda_{m+1}-\lambda_m}\right)\int_0^t \exp((\lambda_{m+2}-\lambda_m)u)\mathrm{d}u\right. \\
&\left.+\left(\frac{1}{\lambda_m-\lambda_{m+1}}\right)\int_0^t \exp((\lambda_{m+2}-\lambda_{m+1})u)\mathrm{d}u\right\} \\
&= \exp(-\lambda_{m+2}t)\lambda_{m+1}\lambda_m\left\{\left(\frac{1}{\lambda_{m+1}-\lambda_m}\right)\left(\frac{\exp(\lambda_{m+2}t-\lambda_m t)-1}{\lambda_{m+2}-\lambda_m}\right)\right. \\
&\left.+\left(\frac{1}{\lambda_m-\lambda_{m+1}}\right)\left(\frac{\exp(\lambda_{m+2}t-\lambda_{m+1}t)-1)}{\lambda_{m+2}-\lambda_{m+1}}\right)\right\} \\
&= \left\{\prod_{j=0}^{1}\lambda_{m+j}\right\}\exp(-\lambda_{m+2}t)\left\{\frac{(\exp(\lambda_{m+2}t-\lambda_m t)-1)}{(\lambda_{m+1}-\lambda_m)(\lambda_{m+2}-\lambda_m)}\right. \\
&\left.+\frac{(\exp(\lambda_{m+2}t-\lambda_{m+1}t)-1)}{(\lambda_m-\lambda_{m+1})(\lambda_{m+2}-\lambda_{m+1})}\right\} \\
&= \left\{\prod_{j=0}^{1}\lambda_{m+j}\right\}\left\{\frac{\exp(-\lambda_m t)-\exp(-\lambda_{m+2}t)}{(\lambda_{m+1}-\lambda_m)(\lambda_{m+2}-\lambda_m)}\right. \\
&\left.+\frac{\exp(-\lambda_{m+1}t)-\exp(-\lambda_{m+2}t)}{(\lambda_m-\lambda_{m+1})(\lambda_{m+2}-\lambda_{m+1})}\right\} \\
&= \left\{\prod_{j=0}^{1}\lambda_{m+j}\right\}\sum_{j=0}^{1}\frac{\exp(-\lambda_{m+j}t)-\exp(-\lambda_{m+2}t)}{\prod_{l=0,l\neq j}^{2}(\lambda_{m+l}-\lambda_{m+j})}. \qquad (2.11)
\end{aligned}$$

For any $n \in \mathbb{N}$, we can confirm that the following equation holds:

$$\begin{aligned}
&Q_n(t) \\
&= \left\{\prod_{j=0}^{n-1}\lambda_{m+j}\right\}\sum_{j=0}^{n-1}\frac{\exp(-\lambda_{m+j}t)-\exp(-\lambda_{m+n}t)}{\prod_{l=0}^{n,l\neq j}(\lambda_{m+l}-\lambda_{m+j})} \\
&= \left\{\prod_{j=0}^{n-1}\lambda_{m+j}\right\}\left\{\sum_{j=0}^{n-1}\frac{\exp(-\lambda_{m+j}t)}{\prod_{l=0,l\neq j}^{n}(\lambda_{m+l}-\lambda_{m+j})}-\sum_{j=0}^{n-1}\frac{\exp(-\lambda_{m+n}t)}{\prod_{l=0,l\neq j}^{n}(\lambda_{m+l}-\lambda_{m+j})}\right\}. \qquad (2.12)
\end{aligned}$$

Since the above expression is a little complicated, we try to get a simpler expression. Following Gat [42], it holds that

$$\sum_{j=0}^{n}\frac{1}{\prod_{l=0,l\neq j}^{n}(\lambda_{m+l}-\lambda_{m+j})} = 0. \qquad (2.13)$$

Finally, we can derive the general solution of the conditional probability $Q_n(t)$ in Equation (2.6). Based on the result, the steady-state transition probabilities

in M/G&S-SRMs are given by

$$Q_n(t) = \sum_{j=0}^{n} \frac{k^{\frac{n(n-1)}{2}} \exp(-bk^{m+j}t)}{\prod_{l=0,l\neq j}^{n}(k^l - k^j)}, \tag{2.14}$$

and

$$Q_n(t) = \sum_{j=0}^{n} \frac{\exp\left[-\left(b\exp(-(m+j)at) + \frac{n(n-1)}{2}a\right)\right]}{\prod_{l=0,l\neq j}^{n}\left[\exp\left(-la\right) - \exp\left(-ja\right)\right]}, \tag{2.15}$$

respectively. When $s = 0$ and $m = 0$, we have $Q_n\left(t\right) = P_n\left(t\right) = \Pr[N\left(t\right) = n|N(0) = 0]$, which is known as the p.m.f..

Once the p.m.f. $P_n(t)$ is obtained, the mean value function $\mathrm{E}[N(t)]$ can be calculated numerically. Boland and Singh [35] corrected a mistake on the mean value function in M-SRM in the standard textbook [2], and suggested applying the probability generating function $P(s,t) = \sum_{n=0}^{\infty} P_n(t)s^n$ with t (> 0) and $s \in (0,1)$. We also use their result directly to obtain the mean value functions for M/G&S-SRMs;

$$\mathrm{E}\left[N\left(t\right)\right] = bt + \sum_{j=2}^{\infty}(-1)^{j-1}\frac{(bt)^j}{j!}\prod_{n=1}^{j-1}(1-k^n), \tag{2.16}$$

$$\mathrm{E}\left[N(t)\right] = bt + \sum_{j=2}^{\infty}(-1)^{j-1}\frac{(bt)^j}{j!}\prod_{n=1}^{j-1}\left[1-\exp\left(-an\right)\right], \tag{2.17}$$

respectively. It is worth mentioning that the above expressions are based on infinite series. Since the computation of Equations (2.16) and (2.17) is rather unstable, we need to evaluate the termwise calculation with verified computation carefully.

2.3 Parameter Estimation

Suppose that software fault count group data are available and consist of a pair of the time interval from $\tau = 0$ and the cumulative number of software faults; $\left(\tau_i,\ n_i\right), i = 1,\ 2,\ \ldots m$. Then the likelihood function is given as the product of conditional probabilities $Q_i(\tau)$ $(i = 1, 2, \ldots, m)$ by

$$\mathcal{L}\left(\boldsymbol{\nu}\right) = \prod_{i=1}^{m} P[N\left(\tau_i\right) - N\left(\tau_{i-1}\right) = n_i - n_{i-1}|N(s) = n_{i-1}], \tag{2.18}$$

where $\boldsymbol{\nu}$ is a free parameter vector involved in the failure detection rate. From Equation (2.2), the log-likelihood function for J&M-SRM $\nu = (N, b)$ is given by

$$\ln \mathcal{L}(\boldsymbol{\nu}) = \sum_{i=1}^{m} \ln\Big\{\Big(N - n_{i-1}\Big)! - (n_i - n_{i-1})! - \big(N - n_i\big)!\Big\} - \sum_{i=1}^{m} \left((N - n_i)\, a\, (\tau_i - \tau_{i-1})\right) + \sum_{i=1}^{m} \Big\{(n_i - n_{i-1}) \ln\Big[1 - e^{-a(\tau_i - \tau_{i-1})}\Big]\Big\}. \tag{2.19}$$

The maximum likelihood estimate of model parameter $\boldsymbol{\nu}$ is given by the maximize $\hat{\boldsymbol{\nu}}$ for $\ln \mathcal{L}(\boldsymbol{\nu})$.

For the M/G&S-SRMs, we obtain the log-likelihood functions:

$$\ln \mathcal{L}(\boldsymbol{\nu}) = \sum_{i=1}^{m} \ln \left\{ \sum_{j=0}^{n_i - n_{i-1}} (-1)^j \exp\left[-b(\tau_i - \tau_{i-1}) k^{(n_{i-1}+j)}\right] \prod_{l=1}^{j} \left(\frac{1}{k^l - 1}\right) \times \prod_{l=1}^{n_i - n_{i-1} - j} \left(\frac{k^{(l-1)}}{k^l - 1}\right) \right\} \tag{2.20}$$

and

$$\ln \mathcal{L}(\boldsymbol{\nu}) = \sum_{i=1}^{m} \left\{ -\frac{(n_i - n_{i-1})\,(n_i - n_{i-1} - 1)}{2} a + \ln \left[\sum_{j=0}^{n_i - n_{i-1}} \frac{\exp\left[-b\,(\tau_i - \tau_{i-1}) \exp\left(-(n_{i-1} + j)a\right)\right]}{\prod_{l=0, l\neq j}^{n_i - n_{i-1}} \left[\exp(-la) - \exp(-ja)\right]} \right] \right\}, \tag{2.21}$$

for $\boldsymbol{\nu} = (b, k)$ and $\boldsymbol{\nu} = (a, b)$, respectively. Hence, the problem is to derive the optimal $\boldsymbol{\nu}$ by maximizing $\ln \mathcal{L}(\boldsymbol{\nu})$ with $\boldsymbol{\nu} = (b, k)$ or $\boldsymbol{\nu} = (a, b)$.

2.4 Numerical Experiments

In our numerical experiments, we use a total of eight data sets of software fault count group (time-interval) data; $(\tau_i,\ n_i)\,, i = 1,\ 2,\ \ldots m$, which were collected from actual software development projects, where n_i is the cumulative number of faults detected by each time point t_i. The data sources and features of the target software systems are summarized in Table 1.1 (b). In this chapter, we re-name them from GDS1 to GDS8.

2.4.1 Goodness-of-fit Performance

First, we attempt to investigate the goodness-of-fit performance of the J&M-SRM and M/G&S-SRM for the group data in Table 1.1. In Figure2.2, we depict the behavior of the cumulative number of software faults in GDS1 and the mean value functions of J&M-SRM and M/G&S-SRM. From this figure, it is not easy to find out the remarkable difference between the two de-eutrophication SRMs. For the model selection, we apply the Akaike information criterion (AIC) and the mean squares error (MSE). Once the maximum log-likelihood $\ln \mathcal{L}(\hat{\boldsymbol{\nu}})$ is given with the maximum likelihood estimate $\hat{\boldsymbol{\nu}}$, AIC is defined by

$$\mathrm{AIC} = -2 \times \ln L\left(\hat{\boldsymbol{\nu}}\right) \ + 2 \times (\text{the number of parameters}). \tag{2.22}$$

Since AIC is an approximate distance between the assumed SRM and the real (but unknown) SRM behind the underlying data, the smaller AIC implies the better SRM. As an alternative goodness-of-fit measure, we define MSE, which is a vertical distance between the assumed SRM and the underlying data;

$$\mathrm{MSE} = \frac{\sum_{i=1}^{m} \{n_i - \mathrm{E}\left[N(\tau_i); \hat{\boldsymbol{\nu}}\right]\}^2}{m}. \tag{2.23}$$

Of course, the smaller MSE is the better SRM in terms of the goodness-of-fit to the underlying data. In Table 2.1, we compare J&M-SRM with M/G&S-SRM in terms of AIC, where the case with the minimum AIC is marked with bold font in each data set. From these results, it is seen that M/G&S-SRM outperformed in GDS5, GDS6, GDS7, and GDS8, but J&M-SRM was better in GDS2, GDS4, and GDS6. The important thing is that the differences between J&M-SRM and M/G&S-SRM are not so significant in terms of AIC in GDS1, GDS2, GDS3, GDS4, GDS5, and GDS6, because the differences are at most 2 in AIC. However, in GDS7 and GDS8, M/G&S-SRM gave rather smaller AICs than J&M-SRM. Looking at MSEs, the differences between two SRMs seem to be large, but we confirm that SRM with smaller AIC provides smaller MSE as well.

2.4.2 Predictive Performance

Next, we investigate the predictive performances of J&M-SRM and M/G&S-SRM with group data. In our experiment, we set three observation points;

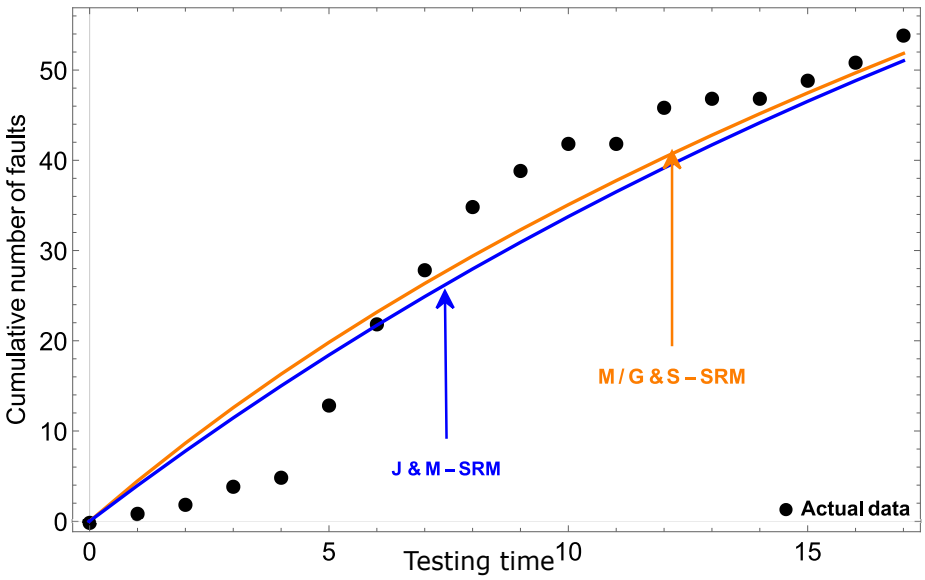


Figure 2.2: Behavior of the cumulative number of software faults detected in GDS1.

Table 2.1: Goodness-of-fit performances based on AIC/MSE

	J&M-SRM		M/G&S-SRM	
	AIC	MSE	AIC	MSE
GDS1	**86.339**	**25.432**	87.139	30.181
GDS2	**61.393**	**53.018**	61.579	53.317
GDS3	**62.521**	**28.620**	65.437	63.684
GDS4	**29.323**	**0.339**	29.440	0.356
GDS5	110.498	59.943	**109.679**	**33.262**
GDS6	141.905	51.109	**141.088**	**43.316**
GDS7	174.011	176.123	**114.129**	**91.165**
GDS8	190.336	232.109	**159.063**	**116.366**

20%, 50%, and 80% points of the whole data set, and are interested in examining the prediction ability of the SRMs in the early, middle and late software testing phases. Figures 2.3, 2.4 and 2.5 show the prediction results with GDS1 from 30%, 50%, 80% observation points. In Figures 2.3 $\sim$ 2.5, we can observe that the prediction result of M/G&S-SRM is better than J&M-SRM, but the difference seems to be slightly small. For a more precise comparison of the predictive performances, we calculate the predictive mean squares error (PMSE) and the predictive log-likelihood (PLL). Suppose that $(\tau_i,\ n_i),\ i = 1,\ 2,\ \ldots l$ are observed at the observation point (τ_l, n_l). For all the data set $(\tau_i,\ n_i), i = 1,\ 2,\ \ldots m$ $(l < m)$, PMSE is given by

$$\mathrm{PMSE} = \sum_{i=l+1}^{m} \left(\mathrm{E}\left[N\left(\tau_i\right);\ \hat{\boldsymbol{\nu}}\right] - n_i\right)^2 / (m-l). \tag{2.24}$$

The smaller PMSE means the better SRM. On the other hand, PLLs are derived as

$$\begin{aligned}\mathrm{PLL}\left(\hat{\boldsymbol{\nu}}\right) = &\\ &\sum_{i=l+1}^{m} \ln\left\{\left(\hat{N} - n_{i-1}\right)! - (n_i - n_{i-1})! - \left(\hat{N} - n_i\right)!\right\}\\ &- \sum_{i=l+1}^{m} \left(\left(\hat{N} - n_i\right)\theta\left(\tau_i - \tau_{i-1}\right)\right) + \sum_{i=l+1}^{m} \left\{(n_i - n_{i-1}) \ln\left[1 - e^{-\hat{a}(\tau_i - \tau_{i-1})}\right]\right\},\end{aligned} \tag{2.25}$$

$$\begin{aligned}\mathrm{PLL}\left(\hat{\boldsymbol{\nu}}\right) = &\\ &\sum_{i=l+1}^{m} \ln\left\{\sum_{j=0}^{n_i - n_{i-1}} (-1)^j \exp\left[-\hat{b}(\tau_i - \tau_{i-1})\hat{k}^{(n_{i-1}+j)}\right] \prod_{s=1}^{j} \left(\frac{1}{\hat{k}^s - 1}\right)\right.\\ &\left.\times \prod_{s=1}^{n_i - n_{i-1} - j} \left(\frac{\hat{k}^{(s-1)}}{\hat{k}^s - 1}\right)\right\},\end{aligned} \tag{2.26}$$

for J&M-SRM and M/G&S-SRM, respectively, where $\hat{\boldsymbol{\nu}} = (\hat{N}, \hat{a})$ and $\hat{\boldsymbol{\nu}} = (\hat{b},\ \hat{k})$ are the maximum likelihood estimates with $(\tau_i,\ n_i), i = 1,\ 2,\ \ldots l$. The larger PLL means the better SRM in teams of prediction.

In the comparison based on PMSE and PLL, it is found that J&M-SRM over-estimated the future trend of the cumulative number of software faults, but M/G&S-SRM could make the better prediction of the unknown patterns in the future. In Tables 2.2, 2.3 and 2.4, we compare J&M-SRM with M/G&S-SRM in terms of PMSE and PLL. In the early testing phase at 20% observation

point, M/G&S-SRM could outperform in 6 out of 8 data sets, but J&M-SRM could make the better prediction in only GDS2 and GDS5 from the viewpoints of PMSE and PLL. In the middle testing phase with 50% observation point, M/G&S-SRM gave the smaller PMSE (larger PLL) in 6 (7) data sets except in GDS5 and GDS8 (GDS5). In the later testing phase at 80% observation point, M/G&S-SRM provided the smaller PMSE and larger PLL in 5 data sets except in GDS2, GDS5, and GDS8. The lesson learned from the experiment suggests that geometric de-eutrophication SRM (M/G&S-SRM) could outperform the J&M-SRM in terms of predictive performances in many cases.

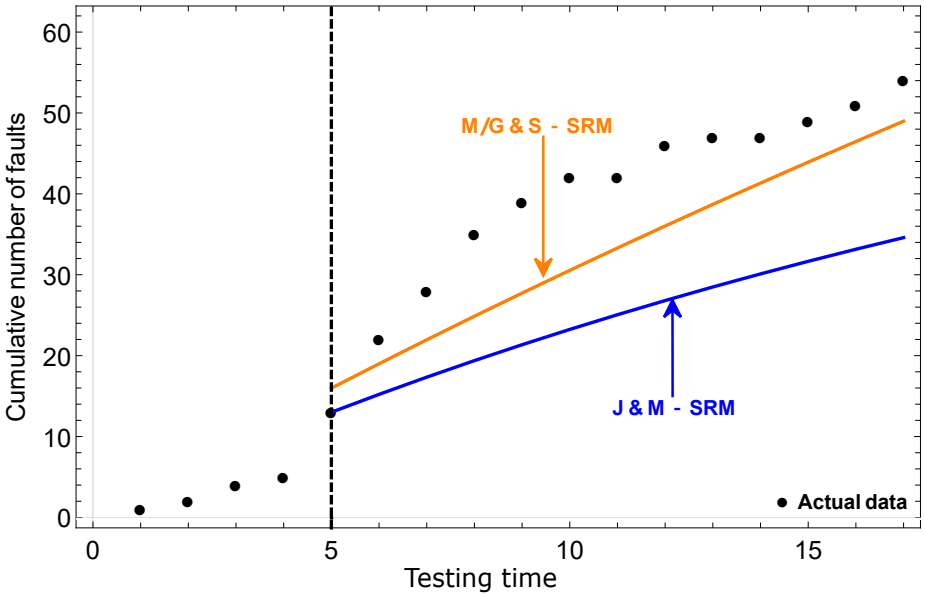


Figure 2.3: Behavior of the cumulative number of software faults detected in GDS1 (20% observation point).

2.4.3 Software Reliability Assessment

Finally, we concern to quantify the software reliability. Let $R(x)$ denote the software reliability, which is the probability that the software is fault-free in the time interval $(t, t+x]$, where x (> 0) is an operational time interval after the release (prediction length);

$$R(x) = \Pr\{N(t+x) - N(t) = 0 | N(t) = n\}. \tag{2.27}$$

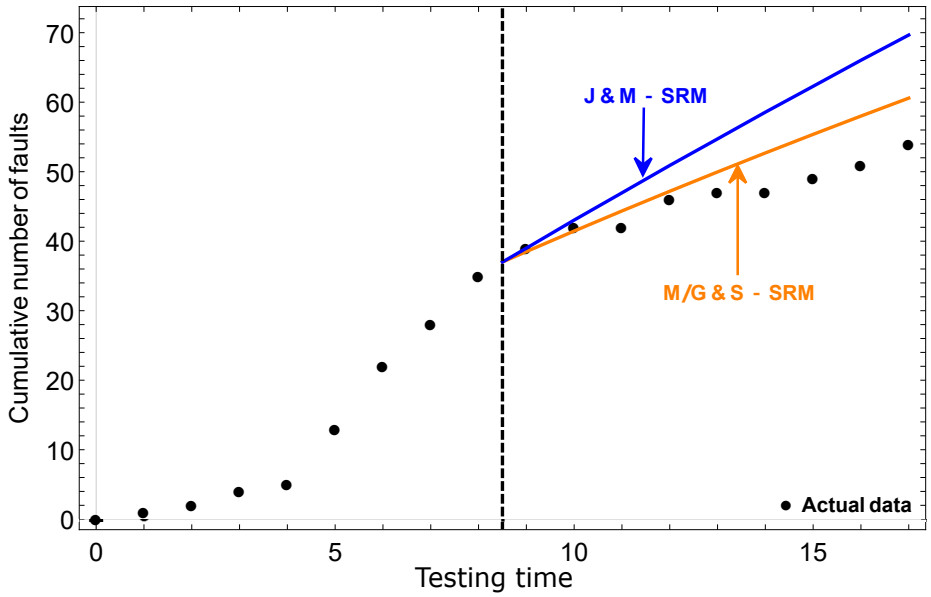


Figure 2.4: Behavior of the cumulative number of software faults detected in GDS1 (50% observation point).

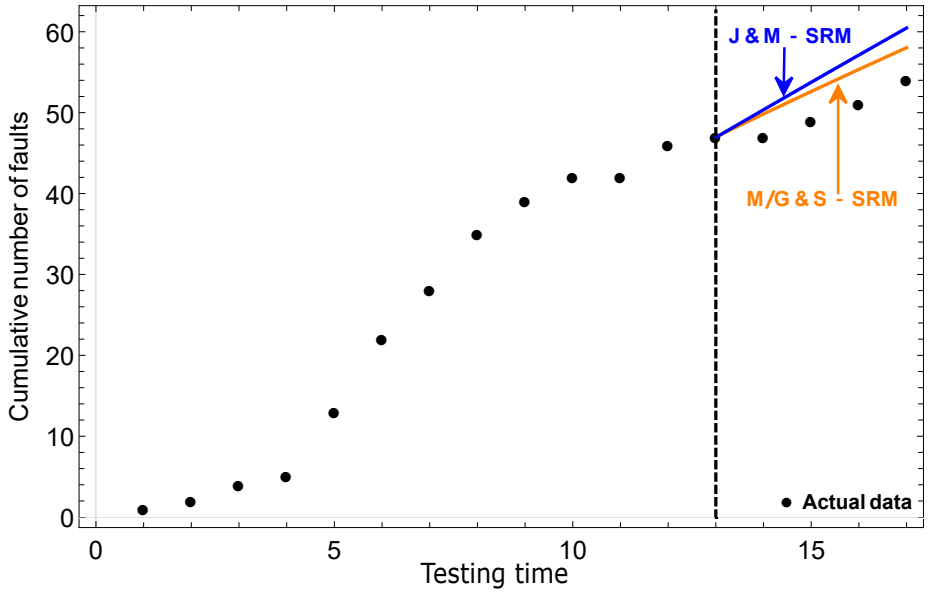


Figure 2.5: Behavior of the cumulative number of software faults detected in GDS1 (80% observation point).

Table 2.2: Comparison of PMSE/PLL with de-eutrophication SRMs (20% observation point).

	J&M-SRM		M/G&S-SRM	
	PMSE	PLL	PMSE	PLL
GDS1	296.612	-31.845	**62.819**	**-24.181**
GDS2	**32.027**	**-16.354**	57.589	-27.863
GDS3	1263.475	-124.200	**227.240**	**-4.978**
GDS4	5.268	-12.553	**0.459**	**-8.937**
GDS5	**414.531**	**-96.417**	802.693	-138.406
GDS6	513.294	-113.024	**450.582**	**-70.066**
GDS7	1139.899	-106.504	**1005.682**	**-105.597**
GDS8	184.676	-77.006	**160.628**	**-48.841**

Table 2.3: Comparison of PMSE/PLL with de-eutrophication SRMs (50% observation point).

	J&M-SRM		M/G&S-SRM	
	PMSE	PLL	PMSE	PLL
GDS1	90.149	-22.282	**36.942**	**-16.548**
GDS2	58.912	-199.399	**9.389**	**-14.171**
GDS3	1000.483	-18.869	**236.564**	**-13.891**
GDS4	1.141	-6.300	**0.419**	**-6.123**
GDS5	**400.128**	**-85.149**	967.386	-131.751
GDS6	80.156	-34.379	**64.218**	**-33.706**
GDS7	331.748	-33.7145	**22.942**	**-21.911**
GDS8	**50.536**	-46.7612	146.080	**-33.137**

Table 2.4: Comparison of PMSE/PLL with de-eutrophication SRMs (80% observation point).

	J&M-SRM		M/G&S-SRM	
	PMSE	PLL	PMSE	PLL
GDS1	37.717	-8.368	**5.991**	**-7.115**
GDS2	**3.116**	**-4.370**	18.614	-6.123
GDS3	570.037	-10.308	**68.933**	**-6.123**
GDS4	2.347	-3.573	**1.117**	**-2.971**
GDS5	**6.167**	**-7.348**	64.781	-11.292
GDS6	41.702	-12.387	**8.373**	**-9.806**
GDS7	30.209	-10.065	**19.773**	**-6.955**
GDS8	**12.904**	**-12.538**	33.019	-12.648

For the release time t_l, since the time intervals between l-th and $(l+1)$-st fault-detection time is exponentially distributed with parameters $\hat{b}(\hat{N}-1)$ for J&M-SRM and $\hat{b}\hat{k}^l$ for M/G&S-SRM, the software reliability functions for the respective de-eutrophication SRMs are given by

$$R(x) = \exp(-\hat{b}(\hat{N}-l+1)x), \tag{2.28}$$

$$R(x) = \exp(-\hat{b}\hat{k}^l x). \tag{2.29}$$

In our experiment, we set the prediction length x as the exact same testing time measured by calendar time in each group data. Table 2.5 presents the quantitative software reliability with the group data sets, where we denote the larger software reliability value with the bold font. It is seen that J&M-SRM provided the larger software reliability values in 6 data sets. In other words, the well-known J&M-SRM tends to give the more optimistic decision in software reliability evaluation. If we can suppose that all the projects succeeded and no software faults were reported after the release, J&M-SRM seems to be more reliable than M/G&S-SRM in software release decisions. However, it is worth mentioning that the resulting software reliability values are all small. This implies that all the software products should continue being tested further.

So, we can conclude that in this chapter, we have performed the group data analysis for a de-eutrophication SRM based on a pure birth process and

Table 2.5: Comparison of software reliability.

	J&M-SRM	M/G&S-SRM
GDS1	**1.576E-17**	2.516E-20
GDS2	**1.553E-06**	5.112E-09
GDS3	1.272E-15	**1.037E-14**
GDS4	**2.756E-03**	2.275E-03
GDS5	**1.655E-10**	8.191E-12
GDS6	6.693E-23	**3.401E-22**
GDS7	**5.059E-17**	8.919E-21
GDS8	**9.621E-33**	3.345E-36

compared it with the well-known J&M-SRM in terms of goodness-of-fit and predictive performances. As we have already emphasized, the group data analysis for a de-eutrophication SRM had been left in the software reliability research for a long time. In numerical examples with 8 actual software development project data sets, we have shown that the geometric de-eutrophication SRM was much more attractive to make the software reliability prediction, although the seminal J&M-SRM based on the linear death process has been used more frequently.

Chapter 3

Extension of NHPP-based SRMs

Among the existing SRMs, the NHPP-based SRMs are recognized as an essential class because of their mathematical tractability and high applicability, and have been widely used to describe the behavior of the cumulative number of software faults. Almost all representative existing NHPP-based SRMs are developed based on the finite-failure assumption and are characterized by a mean value function that is proportional to the cumulative distribution function (c.d.f.) of the software fault-detection time. But, only a few NHPP-based SRMs have also been proposed under the infinite-failure assumption. It is worth noting that the c.d.f.s are the representative lifetime distribution functions to model the time to failure in reliability engineering. On one hand, up to the present stage, we have known that no unique SRM, which could fit every software fault count data, was found yet, and that the best SRM strongly depended on the kind of software fault count data. Hence, in this chapter, we have two research questions; "Are there some novel time distribution families that are more applicable to describe software fault detection times?" and "Are infinite-failure NHPP-based SRMs really useful?". More specifically, we developed 11 infinite-failure (type-II) NHPP-based SRMs by introducing some representative software fault-detection time distributions (e.g., generalized exponential distributions family, extreme-value distribution family) into the infinite-failure assumption. On the other hand, we introduce the Burr-type and Lindley-type distributions into NHPP-based software reliability modeling, and develop several finite-failure (type-I) and infinite-failure

(type-II) NHPP-based SRMs. We compare our proposed SRMs with the existing NHPP-based SRMs in terms of goodness-of-fit and predictive performances.

3.1 NHPP-based Software Reliability Modeling

As we know, most textbooks [1, 2, 3] have pointed out that when the mean value function was used to characterize the cumulative number of software failures by time t, there were two types of NHPP-based SRMs; *finite-failure (type-I) NHPP-based SRMs* and *infinite-failure (type-II) NHPP-based SRMs*.

3.1.1 Existing Type-I NHPP-based SRMs

In the software reliability modeling framework under the type-I NHPP assumption, before the testing, the remaining number of software faults is assumed to obey a Poisson distribution with a positive mean ω. Each software fault is assumed to be detected at independent and identically distributed (i. i. d.) random time, and is fixed immediately just after it is detected. For any $t \in (0, +\infty)$, $F(t; \boldsymbol{\alpha})$, a non-decreasing function, is applied to describe the time distribution of each fault detection during the software testing phase, which is also known as the c.d.f. In the expression, $\boldsymbol{\alpha}$ indicates the free parameter vector in the c.d.f. Then, a binomial distributed random variable with probability $F(t; \boldsymbol{\alpha})$ with a Poisson distributed population with parameter ω is employed to characterize the resultant software fault-detection process. From a simple algebraic manipulation, the mean value function of NHPP can be derived as

$$M(t; \boldsymbol{\theta}) = \omega F(t; \boldsymbol{\alpha}), \tag{3.1}$$

which can also be recognized as the cumulative number of faults detected by the software testing at time point t with $\boldsymbol{\theta} = (\omega, \boldsymbol{\alpha})$ and $\lim_{t\to\infty} M(t; \boldsymbol{\theta}) = \omega$ (> 0). This property is consistent with the assumption of software reliability modeling for type-I NHPP in which the number of initial remaining faults before the software testing is finite. The best-known type-I NHPP-based SRM was proposed by Goel and Okumoto [10], where they assumed the exponential distribution as the fault-detection time distribution in the software testing. The mean value function there is in proportion to the cumulative distribution function (c.d.f.) of the exponential distribution. After that, by postulating the other fault-detection

time distributions, several type-I NHPP-based SRMs have been proposed in the literature, such as, the truncated-normal NHPP-based SRM [17], the log-normal NHPP-based SRM [8, 17], the truncated-logistic NHPP-based SRM [15], the log-logistic NHPP-based SRM [13], the extreme-value NHPP-based SRMs [12, 16], the gamma NHPP-based SRM [19, 20], and the Pareto NHPP-based SRM [7]. In Table 3.1, we summarize 11 existing type-I NHPP-based SRMs with their associated c.d.f.s and bounded mean value functions, which were employed in the software reliability assessment tool on the spreadsheet (SRATS) by Okamura and Dohi [43].

Even though the type-I NHPP-based SRMs are recognized as plausible models in terms of software reliability growth phenomena, it has to be acknowledged that reliability engineers sometimes feel discomfort when handling the type-I NHPPs, since the inter-failure time distributions in the type-I NHPP-based SRMs are *defective* [44]. Let us suppose that the random variables $T_1,\ T_2, ...,\ T_n$ represent the first, second, ..., n-th failure times that occurred since the software testing starts at $T_0 = 0$. Let the random variables, $X_1,\ X_2, ...,\ X_n$, denote the inter-failure times between two consecutive failures;

$$T_n = \sum_{j=1}^{n} X_j = T_{n-1} + X_n, \quad n = 0, 1, 2, ... \tag{3.2}$$

From Equations (1.11) and (3.2), the c.d.f. of T_n can be obtained as

$$\begin{aligned} G_n(t;\boldsymbol{\theta}) &= P\{T_n \le t \text{ (the } n\text{-th failure occurs up to } t)\} \\ &= P\{N_t \ge n \text{ (at least } n \text{ failures occur before time } t)\} \\ &= \int_0^t \frac{\lambda(x;\boldsymbol{\theta})\,[M(x;\boldsymbol{\theta})]^{n-1}}{(n-1)!} \exp\left(-M(x;\boldsymbol{\theta})\right) dx \\ &= \sum_{j=n}^{\infty} \frac{[M(t;\boldsymbol{\theta})]^j}{j!} \exp\left(-M(t;\boldsymbol{\theta})\right) \\ &= 1 - \sum_{j=0}^{n-1} \frac{[M(t;\boldsymbol{\theta})]^j}{j!} \exp\left(-M(t;\boldsymbol{\theta})\right). \end{aligned} \tag{3.3}$$

Then, it is straightforward to see in the type-I NHPP-based SRMs that $\lim_{t\to\infty} G_n(t;\boldsymbol{\theta}) < 1$ for an arbitrary n. In other words, even if the testing time tends to be infinite, there still exists a positive probability of the n-th failure not occurring. It is obvious that the c.d.f. of T_n is defective. Similarly, for realizations of T_i $(i = 1, 2, ..., n)$, $t_1,\ t_2, ..., t_n$, we can obtain the c.d.f. of the

inter-failure time X_n in the time interval $(t_{n-1},\ t_{n-1}+x)$ as follows.

$$\begin{aligned} F_n(x;\boldsymbol{\theta}) &= 1 - \Pr\{N(t_{n-1}+x) - N(t_{n-1}) = 0 \mid N(t_{n-1}) = n-1\} \\ &= 1 - \exp\left(-\left(M(t_{n-1}+x;\boldsymbol{\theta}) - M(t_{n-1};\boldsymbol{\theta})\right)\right), \end{aligned} \tag{3.4}$$

where $\Pr\{N(t_{n-1}+x) - N(t_{n-1}) = 0 \mid N(t_{n-1}) = n-1\}$ denotes the probability that no failure occurs in time interval $(t_{n-1},\ t_{n-1}+x)$. Since the mean value function is bounded, i.e., $\lim_{t\to\infty} M(t;\boldsymbol{\theta}) = \omega$, when x is infinite , Equation (3.4) can be reduced to $1 - e^{-(\omega - M(t_{n-1};\boldsymbol{\theta}))} < 1$. It means that regardless of the number of previous failures, the probability that the software fails over an infinite time horizon is always non-zero. Hence, the inter-failure time c.d.f. of type-I NHPP is also defective. For the type-I NHPP-based SRMs, it is not meaningful to discuss some reliability metrics like mean time between failures (MTBF), because the finite moments of T_n and X_n always diverge.

Table 3.1: The representative existing type-I NHPP-based SRMs.

SRM	**Time distribution**	$F(t;\boldsymbol{\alpha})$	$M(t;\boldsymbol{\theta})$
Exp [10]	Exponential distribution	$1-\exp(-\mu_1 t)$	$\omega F(t;\boldsymbol{\alpha})$
Gamma [19],[20]	Gamma distribution	$\int_0^t \frac{\mu_2^{\mu_1} s^{\mu_2-1}\exp(-\mu_2 s)}{\Gamma(\mu_1)} ds$	$\omega F(t;\boldsymbol{\alpha})$
Pareto [7]	Pareto distribution	$1-(\frac{\mu_1}{t+\mu_1})_2^{\mu}$	$\omega F(t;\boldsymbol{\alpha})$
Tnorm [17]	Truncated normal distribution	$\frac{1}{\sqrt{2\pi}\mu_1}\int_{-\infty}^t \exp\left(-\frac{(s-\mu_2)^2}{2\mu_1^2}\right) ds$	$\omega\frac{F(t;\boldsymbol{\alpha})-F(0;\boldsymbol{\alpha})}{1-F(0;\boldsymbol{\alpha})}$
Tlogist [15]	Truncated logistic distribution	$(1-\exp(-\mu_1 t))/(1+\mu_2\exp-\mu_2 t)$	$\omega F(t;\boldsymbol{\alpha})$
Txvmax [16]	Truncated extreme-value max distribution	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$\omega\frac{F(t;\boldsymbol{\alpha})-F(0;\boldsymbol{\alpha})}{1-F(0;\boldsymbol{\alpha})}$
Txvmin[16]	Truncated extreme-value min distribution	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$\omega\frac{F(0;\boldsymbol{\alpha})-F(t;\boldsymbol{\alpha})}{F(0;\boldsymbol{\alpha})}$
Lnorm [8],[17]	Log-normal distribution	$\frac{1}{\sqrt{2\pi}\mu_1}\int_{-\infty}^t \exp\left(-\frac{(s-\mu_2)^2}{2\mu_1^2}\right) ds$	$\omega F(\ln t;\boldsymbol{\alpha})$
Llogist[13]	Log-logistic distribution	$(1-\exp(-\mu_1 t))/(1+\mu_2\exp-\mu_2 t)$	$\omega F(\ln t;\boldsymbol{\alpha})$
Lxvmax [16]	Log-extreme-value max distribution	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$\omega F(\ln t;\boldsymbol{\alpha})$
Lxvmin [12]	Log-extreme-value min distribution	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$\omega\left(1-F(-\ln t;\boldsymbol{\alpha})\right)$

($\omega>0, \mu_1>0, \mu_2>0$)

3.1.2 Type-II NHPP-based SRMs with Representative Software Fault-detection Time Distributions

Type-II NHPP assume that a new software fault was not inserted at each software debugging. However, this assumption may be somewhat specific, because the so-called *imperfect debugging* may occur in the actual software testing phases. When the possibility of imperfect debugging is considered, the assumption of finiteness in the type-I NHPP-based SRMs seems to be rather strong. Similar to the classical preventive maintenance modeling [45], if each software failure is *minimally* repaired through the debugging, the mean value function of the software fault-detection process is unbounded and is given by

$$M(t;\boldsymbol{\alpha}) = -\ln(1 - F(t;\boldsymbol{\alpha})), \tag{3.5}$$

where $\lim_{t\to\infty} M(t;\alpha) \to \infty$. It is obvious that the c.d.f.s, $G_n(t;\boldsymbol{\theta})$ and $F_n(x;\boldsymbol{\theta})$ in Equations (3.3) and (3.4) are not defective, say, $\lim_{t\to\infty} G_n(t;\boldsymbol{\theta}) = 1$ and $\lim_{x\to\infty} F_{X_i}(x;\boldsymbol{\theta}) = 1$. Hence, it becomes significant to consider important metrics such as MTBF. In this modeling framework, investigating the residual number of software faults before testing has no significant meaning, because it may increase by imperfect debugging through the software testing.

As far as we know, the Cox-Lewis process [46] is one of the earliest type-II NHPPs. The unbounded mean value function of this model is given by $M(t;\boldsymbol{\alpha}) = \frac{(\exp(\mu_1+\mu_2 t)-\exp(\mu_1))}{c}$ with the extreme-value distribution $F(t;\boldsymbol{\alpha}) = 1 - \exp(\exp(\mu_1 + \mu_2 t) - \exp(\mu_1))/\mu_2$. This distribution is also referred to as truncated extreme-value minimum distribution in [43]. Another well-known type-II NHPP-based SRM is referred to as a power-law process model [11, 47, 48], where mean value function and c.d.f. are given by $M(t;\boldsymbol{\alpha}) = (\mu_2/\mu_1)\ t^{(1/\mu_1)}$ and $F(t;\boldsymbol{\alpha}) = 1 - \exp\left(-\exp\left(-\frac{\mu_2+\ln(t)}{\mu_1}\right)\right)$, respectively. The latter is also recognized as the log-extreme-value minimum distribution in [43]. Besides the above two representative NHPPs, the well-known logarithmic Poisson execution time SRM [2, 14] belongs to the type-II category, too. The mean value function of this model is given by $M(t;\boldsymbol{\alpha}) = \mu_2 \ln((1+t)/\mu_1)$ with the Pareto distribution $F(t;\boldsymbol{\alpha}) = 1 - (\mu_1/(t+\mu_1))^{\mu_2}$ in [43].

Table 3.2: Type-II NHPP-based SRMs.

SRMs & Time Distributions	$F(t;\boldsymbol{\alpha})$ & $M(t;\boldsymbol{\alpha})$
Exp (HPP)	$F(t;\boldsymbol{\alpha}) = 1-\exp(-\mu_1 t)$
(Exponential distribution)	$M(t;\boldsymbol{\alpha}) = \mu_1 t$
Gamma	$F(t;\boldsymbol{\alpha}) = \int_0^t \frac{\mu_2^{\mu_1} s^{\mu_1-1}\exp(-\mu_2 s)}{\Gamma(\mu_1)} ds$
(Gamma distribution)	$M(t;\boldsymbol{\alpha}) = \ln(\Gamma(\mu_1)) - \ln\left(\Gamma\left(\mu_1, \frac{t}{\mu_2}\right)\right)$
Pareto (Musa-Okumoto) [2],[14]	$F(t;\boldsymbol{\alpha}) = 1-(\frac{\mu_1}{t+\mu_1})_2^{\mu}$
(Pareto distribution)	$M(t;\boldsymbol{\alpha}) = -\mu_1(\ln(\mu_2)-\ln(\mu_2+t))$
Tnorm	$F(t;\boldsymbol{\alpha}) = \frac{1}{\sqrt{2\pi}\mu_1}\int_{-\infty}^t \exp\left(-\frac{(s-\mu_2)^2}{2\mu_1^2}\right) ds$
(Truncated normal distribution)	$M(t;\boldsymbol{\alpha}) = \ln\left(\mathrm{erf}\left(\frac{\mu_2}{\sqrt{2}\mu_1}\right)+1\right) - \ln\left(\mathrm{erf}\left(\frac{\mu_2-t}{\sqrt{2}\mu_1}\right)+1\right)$
Lnorm	$F(t;\boldsymbol{\alpha}) = \frac{1}{\sqrt{2\pi}\mu_1}\int_{-\infty}^t \exp\left(-\frac{(s-\mu_2)^2}{2\mu_1^2}\right) ds$
(Log-normal distribution)	$M(t;\boldsymbol{\alpha}) = \ln(2) - \ln\left(\mathrm{erf}\left(\frac{\mu_2-\ln(t)}{\sqrt{2}\mu_1}\right)+1\right)$
Tlogist	$F(t;\boldsymbol{\alpha}) = \frac{1-\exp(-\mu_1 t)}{1+\mu_2\exp(-\mu_2 t)}$
(Truncated logistic distribution)	$M(t;\boldsymbol{\alpha}) = \ln(\exp(\mu_2/\mu_1)+\exp(t/\mu_1)) - \ln(\exp(\mu_2/\mu_1)+1)$
Llogist	$F(t;\boldsymbol{\alpha}) = \frac{1-\exp(-\mu_1 t)}{1+\mu_2\exp(-\mu_2 t)}$
(Log-logistic distribution)	$M(t;\boldsymbol{\alpha}) = \ln\left(\exp(\mu_2/\mu_1)+t^{1/\mu_1}\right) - \frac{\mu_2}{\mu_1}$
Txvmax	$F(t;\boldsymbol{\alpha}) = \exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$
(Truncated extreme-value max distribution)	$M(t;\boldsymbol{\alpha}) = \ln(1-\exp(-\exp(\mu_2/\mu_1))) - \ln\left(1-\exp\left(-\exp\left(\frac{\mu_2-t}{\mu_1}\right)\right)\right)$
Lxvmax	$F(t;\boldsymbol{\alpha}) = \exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$
(Log-extreme-value max distribution)	$M(t;\boldsymbol{\alpha}) = -\ln\left(1-\exp\left(-\exp\left(\frac{\mu_2-\ln(t)}{\mu_1}\right)\right)\right)$
Txvmin (Cox-Lewis) [46]	$F(t;\boldsymbol{\alpha}) = \exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$
(Truncated extreme-value min distribution)	$M(t;\boldsymbol{\alpha}) = -\ln(\exp(-\exp(\mu_2/\mu_1)(\exp(t/\mu_1)-1)))$
Power-law [11],[47],[48]	$F(t;\boldsymbol{\alpha}) = \exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$
(Log-extreme-value min distribution)	$M(t;\boldsymbol{\alpha}) = \mu_2/\mu_1\ t^{(1/\mu_1)}$

3.1.3 Parameter Estimation

Suppose that the total number of faults observed in the testing phase is m by the time observation point t_m, where the time sequence consisting of the time points at which each fault is detected is given by $\boldsymbol{D} = \{t_1, t_2, \ldots, t_m\}$. This kind of time series is called software fault-count time-domain data. Generally, CPU time is used to measure the time-domain data in software testing. Then, from Equation (1.14), for a time-domain data set $\boldsymbol{D} = \{t_1, t_2, \ldots, t_m\}$, the likelihood function of NHPP is as follows.

$$\mathcal{L}(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}; \boldsymbol{D}) = \exp(-M(t_m; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})) \prod_{i=1}^{m} \lambda(t_i; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}). \tag{3.6}$$

Taking logarithm of both sides in Equation (3.6), the log-likelihood function is obtained as

$$\ln\mathcal{L}(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}; \boldsymbol{D}) = \sum_{i=1}^{m} \ln \lambda(t_i; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) - M(t_m; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}). \tag{3.7}$$

The ML estimate, $\hat{\boldsymbol{\theta}}$ or $\hat{\boldsymbol{\alpha}}$, is given by $\text{argmax}_{\boldsymbol{\theta}} \ln\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{I})$ or $\text{argmax}_{\boldsymbol{\alpha}} \ln\mathcal{L}(\boldsymbol{\alpha}; \boldsymbol{I})$.

On the other hand, when the group data $\boldsymbol{I} = \{(t_i, n_i), i = 1, 2, \ldots, m\}$ is available, from Equation (1.19), the likelihood function and log likelihood function of NHPP are given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}; \boldsymbol{I}) = & \exp -[M(t_i; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) - M(t_{i-1}; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})] \\ & \prod_{i=1}^{m} \left[\frac{[M(t_i; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) - M(t_{i-1}; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})]^{n_i - n_{i-1}}}{(n_i - n_{i-1})!} \right],\end{aligned} \tag{3.8}$$

and

$$\begin{aligned}&\ln\mathcal{L}(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}; \boldsymbol{I}) = \\ &\sum_{i=1}^{m} (n_i - n_{i-1}) \ln\{M(t_i; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) - M(t_{i-1}; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})\} - \sum_{i=1}^{m} \ln\{(n_i - n_{i-1})!\} \\ &- M(t_m; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}),\end{aligned} \tag{3.9}$$

respectively. The ML estimate, $\hat{\boldsymbol{\theta}}$ or $\hat{\boldsymbol{\alpha}}$, is given by $\text{argmax}_{\boldsymbol{\theta}} \ln\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{I})$ or $\text{argmax}_{\boldsymbol{\alpha}} \ln\mathcal{L}(\boldsymbol{\alpha}; \boldsymbol{I})$.

3.1.4 Numerical Experients

In our numerical experiments, we select the well-known benchmark software fault count data sets in software reliability engineering, that are observed in mission-critical systems. Although the evolution of these systems may be slower than that of business-oriented systems, effects of failure are much greater. Hence, reliability is particularly important for the developers of these mission-critical systems. In our numerical experiments, we analyzed a total of eight time-domain data sets (DS1 $\sim$ DS8 in Table 2.1 (i)), labeled TDDS1$\sim$TDDS8, and eight group data sets (DS15 $\sim$ DS21 in Table 2.1 (ii)), called TIDS1$\sim$TIDS8.

3.1.4.1 Goodness-of-fit Performances

Suppose that the parameters of SRMs have been estimated by the maximum likelihood estimation. In the first experiment, we employ two criteria for evaluating the goodness-of-fit performance of 11 type-I and type-II NHPP-based

SRMs; AIC and MSE. For time-domain data and group data, the MSE is given by

$$\mathrm{MSE}(\hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}; \boldsymbol{D}) = \frac{\sum_{i=1}^{m} (i - M(t_i; \hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}))^2}{m} \tag{3.10}$$

and

$$\mathrm{MSE}(\hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}; \boldsymbol{I}) = \frac{\sum_{i=1}^{m} (n_i - M(t_i; \hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}))^2}{m}, \tag{3.11}$$

respectively. The AIC with ML estimates generally represents an approximation of the Kullback–Leibler divergence between our proposed SRM and the empirical stochastic process behind the fault count data, while the direct application of MSE exhibits a vertical distance between the estimated mean value function and the fault count data. The smaller the AIC/MSE indicates that the SRM has the better goodness-of-fit performance (the better the fit to the underlying data).

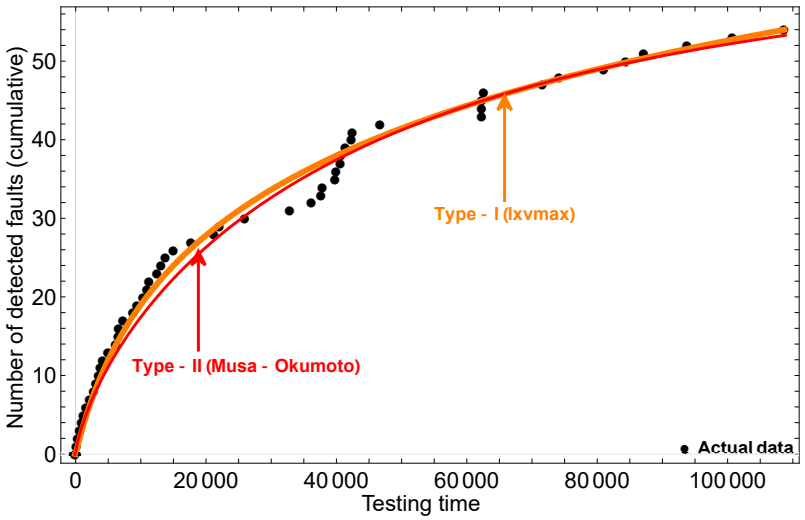


Figure 3.1: Behavior of mean value functions in TDDS1.

Figures 3.1 and 3.2 plot the behavior of the mean value functions of type-I and type-I SRMs in the time-domain data, with TDDS1 and TIDS6. The red curve and the orange curve are plotted as the best SRMs selected from 11 type-II SRMs and 11 type-I SRMs based on their AICs, respectively. Not surprisingly, the two modeling frameworks show slightly different growth trends. More specifically, the type-I (orange curve) always fits better with the actual data in the tail segment, for both the time-domain and group data. However,

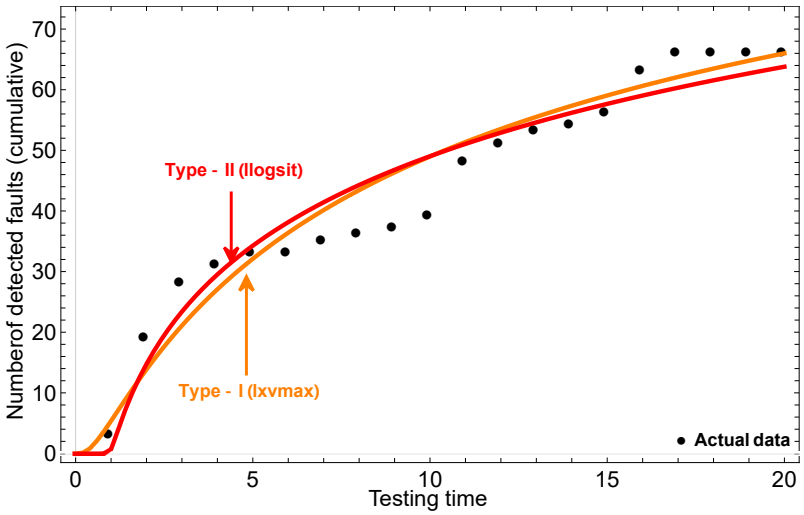


Figure 3.2: Behavior of mean value functions in TIDS6.

we still cannot make a comprehensive assessment of which SRM can exhibit a better fitting ability over the whole data set. This is the reason why we need to think about AIC as well as MSE as the goodness-of-fit criteria. In Table 3.3, we make a more precise comparison between our proposed type-II and the existing type-I on AIC and MSE. Without the comparison from each other, we can find that in the vast majority of cases, the best models in the type-I SRMs are given by the extreme-value distributions. On the contrary, the type-II Pareto (Musa-Okumoto) SRM performs better than the other SRMs. In the next step, by comparing the best type-I and the best type-II SRMs for each data set, it is not difficult to find that in 3 cases (TDDS1, TDDS2 and TDDS5), the type-II SRMs could provide the smaller AIC than the type-I SRMs. However, in all the data sets, the type-I SRMs could provide the smaller MSE than our type-II SRMs. In Table 3.4, we compared the best SRMs of our type-II NHPP with the existing type-I NHPP-based SRMs in 8 group data sets. It can be seen that our type-II SRMs could guarantee the smaller AIC than the existing type-I in 3 cases (TIDS2, TIDS5 and TIDS6), but at the same time, it still cannot outperform the type-I from the viewpoint of MSE for any group data set. We can therefore draw the conclusion that the type-II NHPP-based SRMs could not consistently outperform the existing type-I NHPP-based SRMs in terms of goodness-of-fit performance, but in some cases, especially in the time-domain data, the three

Table 3.3: Goodness-of-fit results in time-domain data.

	Type-I			**Type-II**		
	Best SRM	AIC	MSE	Best SRM	AIC	MSE
TDDS1	Lxvmax	896.666	1.950	Pareto (Musa-Okumoto)	**895.305**	2.315
TDDS2	Lxvmax	598.131	1.705	Pareto (Musa-Okumoto)	**596.501**	1.809
TDDS3	Lxvmin	**1938.160**	6.570	Pareto (Musa-Okumoto)	1939.600	8.052
TDDS4	Txvmin	**759.579**	3.747	Txvmin (Cox-Lewis)	759.948	5.509
TDDS5	Exp	757.869	18.985	Power-law	**757.031**	19.315
TDDS6	Lxvmax	**721.928**	1.442	Txvmin (Cox-Lewis)	726.052	2.803
TDDS7	Lxvmax	1008.220	5.970	Pareto (Musa-Okumoto)	**1007.100**	7.039
TDDS8	Pareto	2504.170	47.404	Pareto (Musa-Okumoto)	**2503.370**	63.699

existing type-II NHPP-based SRMs; Musa-Okumote, Cox-Lewis, and power-law SRMs, could indicate the better experimental results.

Table 3.4: Goodness-of-fit results in group data.

	Type-I			**Type-II**		
	Best SRM	AIC	MSE	Best SRM	AIC	MSE
TIDS1	Llogist	**73.053**	4.115	Tlogist	85.339	48.269
TIDS2	Lxvmax	61.694	3.239	Llogist	**60.674**	3.557
TIDS3	Tnorm	**87.267**	6.151	Txvmin (Cox-Lewis)	91.919	31.232
TIDS4	Tlogist	**51.052**	1.968	Txvmin (Cox-Lewis)	63.556	27.199
TIDS5	Exp	29.911	0.118	Exp	**27.953**	0.186
TIDS6	Lxvmax	108.831	22.514	Llogist	**107.211**	24.394
TIDS7	Txvmin	**123.265**	2.122	Tlogist	138.029	24.847
TIDS8	Llogist	**117.470**	9.408	Llogist	148.438	45.178

3.1.4.2 Predictive Performances

Notably, according to previous studies, it turns out that SRMs with better goodness-of-fit do not necessarily provide excellent predictive performance. In other words, investigating the predictive performance of the type-I and type-II NHPP-based SRMs is of significant importance. Hence, in our second experiment, we employ the PMSE to measure the predictive performance of our type-II

SRMs, where

$$\mathrm{PMSE}(\hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}; \boldsymbol{D}) = \frac{\sum_{i=m+1}^{m+l} \{i - M(t_i; \hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}})\}^2}{l}, \tag{3.12}$$

and

$$\mathrm{PMSE}(\hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}; \boldsymbol{I}) = \frac{\sum_{i=m+1}^{m+l} \{n_i - M(t_i; \hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}})\}^2}{l}, \tag{3.13}$$

for the time-domain and group data respectively. Support that m or n_m software faults have been observed in $(0,\ t_m]$, and the prediction length is given by $l\ (= 1, 2, \cdots)$. $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\alpha}}$ are the ML estimates at observation time t_m for the type-I and type-II NHPP-based SRMs, respectively. Similar to MSE, PMSE is also a metric that evaluates the mean squared distance between the predicted number of detected faults and its (unknown) realization for each prediction length. For a comprehensive investigation of the predictive performance of SRMs at different software testing phases, three observation points were set at 20%, 50% and 80% of the total length of each data set, to represent the early, middle and late phases of software testing and to predict the total number of software faults at the remaining 80%, 50% and 20% of the length of time periods. Then, we calculate the PMSE for the type-I and type-II NHPP-based SRMs. It is immediate to see that the larger the observation point, the shorter the prediction length.

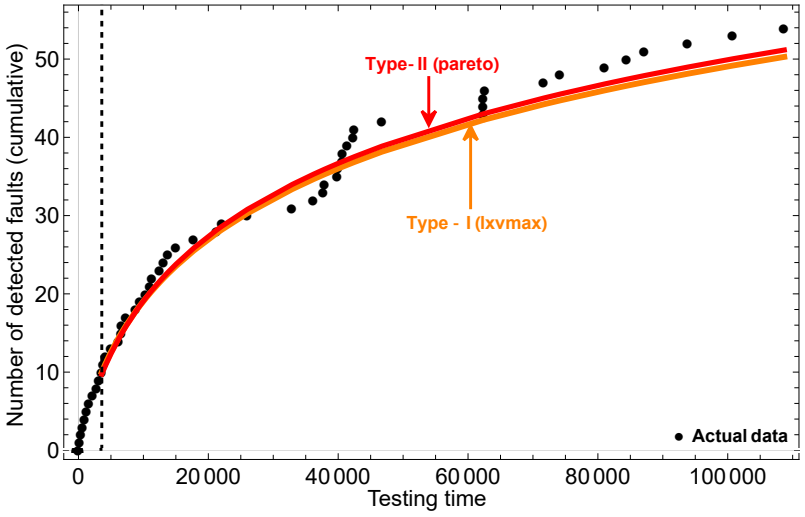


Figure 3.3: Behavior of fault prediction in TDDS1 (20% observation point).

In Figure 3.3 to Figure 3.5, we plot the predictive behavior of the best existing type-I and the best type-II NHPP-based SRMs in time-domain data, TDDS1 at

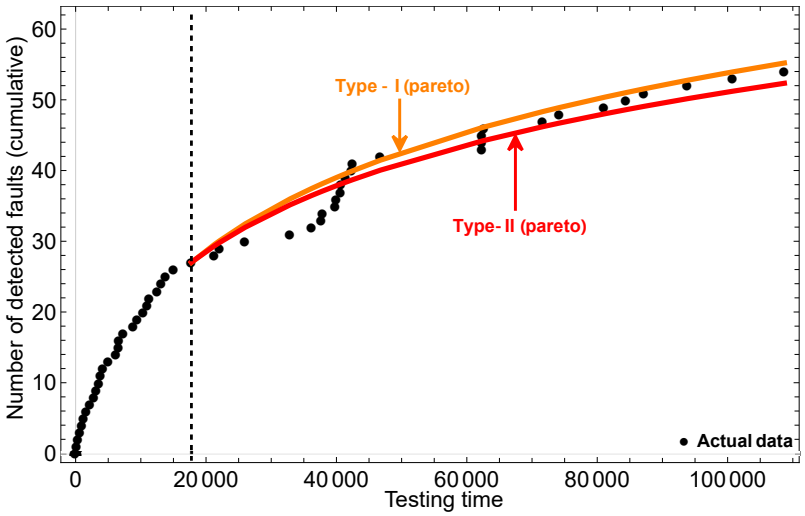


Figure 3.4: Behavior of fault prediction in TDDS1 (50% observation point).

three different observation points. The red curve in each figure represents our best type-II NHPP, while the orange curve denotes the best type-I NHPP. All the best SRMs were taken from the type-I NHPP-based SRMs and the type-II NHPP-based SRMs with their smaller PMSEs in TDDS1. It can be seen that both type-I and type-II tend to give almost the same number of predicted software faults in the early and late testing phases. However, after the mid-term of testing, the type-I NHPP-based SRM tended to make more optimistic software fault predictions. In Figure 3.6 to Figure 3.8, we also plot the predictive behavior of the best existing type-I and the best type-II NHPP-based SRMs in group data, TIDS6. It can be seen that the type-I still tended to miss-predict the number of software faults in the early and middle testing phases. More specifically, in Figure 3.6 and Figure 3.7, while the type-II NHPP-based SRMs could show an increasing trend, the opposite is true for the type-I, whose predictive trend for future phases becomes very flat. However, in Figure 3.8, both the type-I and type-II show more similar predictive trends. In general, prediction of unknown trend changes over longer periods of time in the future is essentially difficult for either the type-I NHPP nor the type-II NHPP. In contrast, the prediction of trend changes over a short period of time is relatively easy, but absolute accuracy cannot be guaranteed.

In Table 3.5, we present the PMSEs of the best type-I SRM compared to

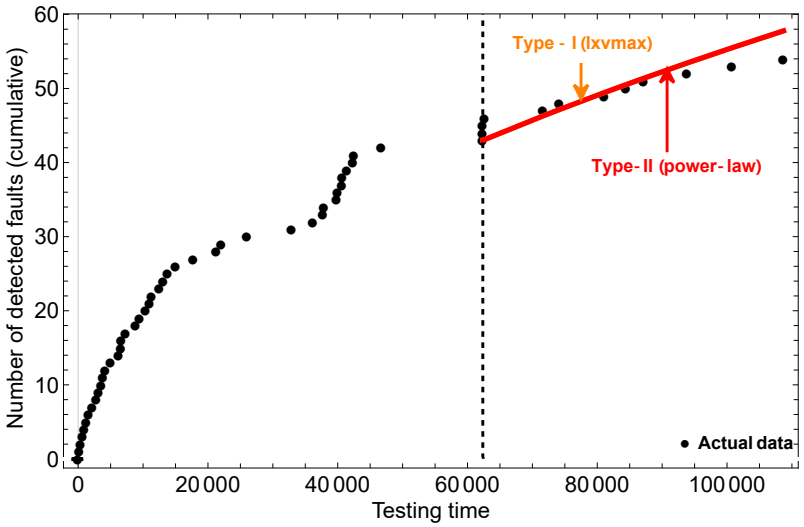


Figure 3.5: Behavior of fault prediction in TDDS1 (80% observation point).

the best type-II SRM in each set of time-domain data. We compare the PMSEs in 11 type-I SRMs and 11 type-II SRMs by selecting the models with their smaller PMSEs as the best SRMs at each observation point. It can be seen that, at 20% observation point, our type-II SRMs could provide the less PMSEs than the existing type-I in 3 cases (TDDS5, TDDS6, and TDDS8). During the middle testing phase (at 50% observation point), we observed that our type-II SRMs outperformed the type-I SRMs in 4 data sets (TDDS3, TDDS7, TDDS5 and TDDS8). As the test proceeded to the late phases (at 80% observation point), the type-II SRMs were able to guarantee the smaller PMSE in TDDS1, TDDS7, and TDDS8. On the other hand, it is found that the best type-II SRMs with better predictive performance than the type-I were the logistic distribution, Musa-Okumoto SRM, and power-law SRM. By comparing PMSE in time-domain data, we believe that the type-II SRMs could become a good alternative to the type-I SRMs. In Table 3.6, when the testing phase is early (at 20% observation point), it can be immediately noticed that our type-II SRMs showed the smaller PMSE than the type-I SRMs in 7 out of 8 group data sets (except in TIDS2). In addition to logistic-based SRM, Musa-Okumoto SRM and power-law SRM, which were proven to perform better in Table 3.5, we observed that Cox-Lewis SRM is also appropriate in some cases of group data (TIDS7 and TIDS8) in terms of predictive performance. At 50% observation

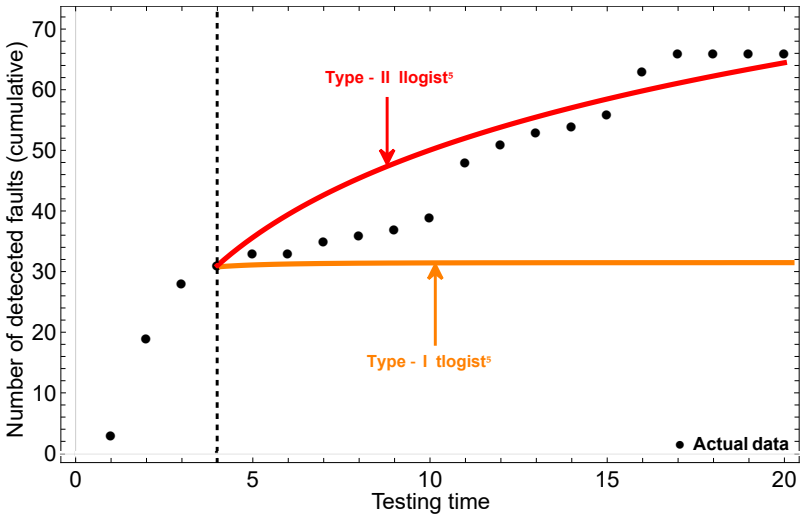


Figure 3.6: Behavior of fault prediction in TIDS6 (20% observation point).

point, we found that the type-II SRMs could guarantee the better predictive performance than the type-I SRMs in 3 cases (TIDS2, TIDS4 and TIDS6). For the late testing phase (at 80% observation point), in only TIDS2, our Tlogist type-II SRMs gave the smallest PMSE in the future prediction phase. In the group data, the predictive performance of the type-II SRMs decreases as the software testing proceeds. Hence, it is possible to summarize that the type-II NHPP-based SRMs outperformed the existing type-I NHPP-based SRMs for software fault detection prediction in the early testing phase when the group data were available.

3.1.4.3 Software Reliability Assessment

Our final research question for NHPP-based type-II SRMs is how to utilize them to quantitatively assess the software reliability. In general, in the NHPP software reliability modeling, the reliability of software at a given time point t_r can be calculated by $R(t_r; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})$, that is, the probability that the software will be failure-free during time interval $(t_m, t_r]$, which can be written as

$$\begin{aligned} R(t_r; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) &= \Pr[N(t_r) - N(t_m) = 0] \\ &= \exp\left(-\left[M(t+x; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) - M(t; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})\right]\right). \end{aligned} \tag{3.14}$$

t_r is defined as the observation point after a certain time of software release, and t_m is the total time for software testing. m is the total number of detected

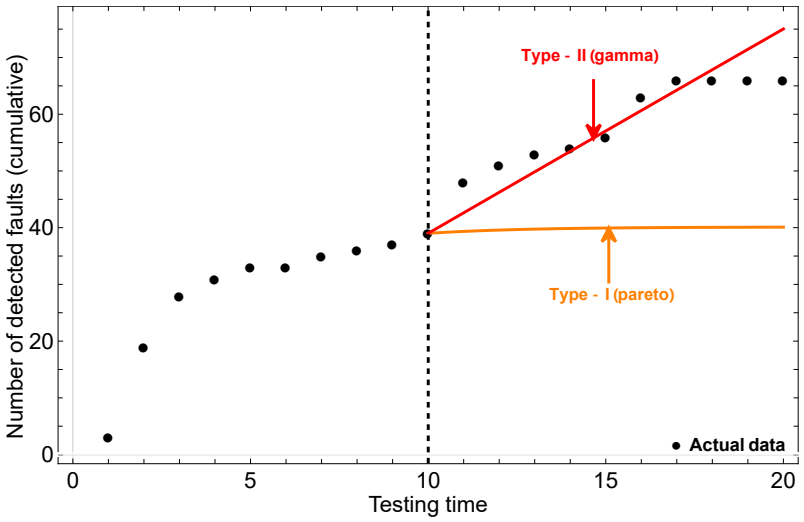


Figure 3.7: Behavior of fault prediction in TIDS6 (50% observation point).

faults before the time point t_m. In this numerical experiment, we assume that $t_r = t_m$. The software reliability in each software development project was predicted quantitatively by importing the mean value functions of type-I NHPP and type-II NHPP into Equation 3.14.

In Table 3.7 and Table 3.8, we compare the quantitative software reliability of the best type-I SRMs and the best type-II SRMs in the time-domain data and group data. We select the type-I SRM and the type-II SRM with their smaller AIC in respective time domain and group data sets as the best SRMs. We can see that in almost all data sets (except in TDDS1 and TIDS6), the type-I SRMs tend to give the higher software reliability than our type-II SRMs. In other words, during the time period $(t_m, t_r]$, the probability of software failure predicted by the type-II NHPP is much higher than that by the type-I NHPP. This observation indicates that our type-II SRMs tend to make more conservative decisions than the type-I SRMs in software reliability assessment. It is important to note that optimistic reliability estimates are often undesirable. This is because software faults are additionally detected as the ex-post results after each observation point in all the data sets.

Finally, we can conclude in this chapter that, under the infinite-failure assumption, in addition to the well-known Musa-Okumoto model, Cox-Lewis model and power-law model, we proposed alternative 8 type-II NHPP-based

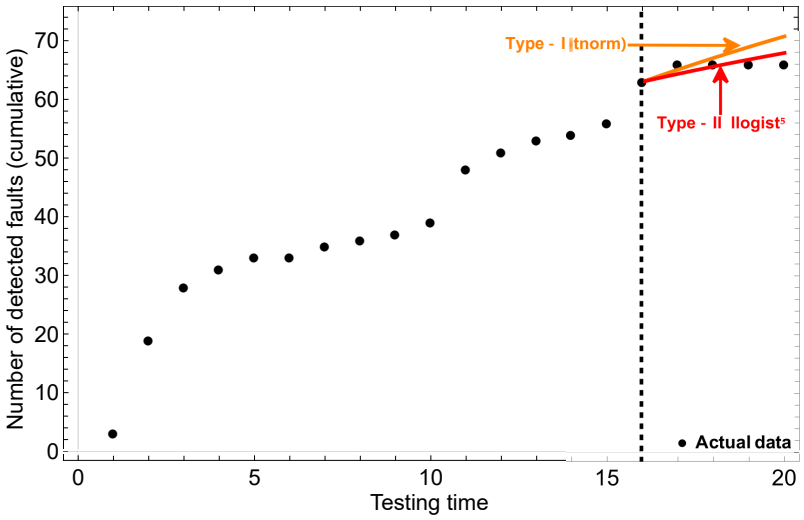


Figure 3.8: Behavior of fault prediction in TIDS6 (80% observation point).

SRMs with 8 different software fault detection time c.d.f.s. By analyzing 8 software fault count time-domain data and 8 software fault count time-interval data (group data), we have investigated the goodness-of-fit performance and predictive performance of our SRMs. We have also compared these SRMs with 11 existing type-I NHPP-based SRMs under the finite-failure assumption. The important point to note is that the type-I and type-II NHPP-based SRMs considered in this chapter have the almost similar software fault detection c.d.f.s. This observation has never been addressed in the past literature.

The experimental results have confirmed that our type-II NHPP-based SRMs could show the better goodness-of-fit performance in some cases. On the other hand, for the group data, the type-II NHPP-based SRMs have exhibited rather better predictive ability than the existing type-I NHPP-based SRMs in the early testing phase. But as the software testing progresses, it has been known that the advantages of type-II NHPP in terms of predictive performance were diminished. Hence, we can conclude that the type-II NHPP-based SRMs cab be considered as the good complements to the type-I NHPP-based SRMs for describing the fault-detection process of software systems. At the same time, they have a greater potential in the early software testing phase. On the other hand, we have also confirmed that the type-II NHPP tended to make more conservative predictions than the type-I NHPP in software reliability assessment.

3.2 Lindley-type NHPP-based Software Reliability Modeling

3.2.1 Lindley-type Distribution

Under Fiducial and Bayesian statistics, a meaningful one-parameter continuous probability distribution called the *Lindley distribution* was proposed by Dennis Victor Lindley [50]. Recently, the Lindley-type distributions have attracted extensive attention. In order to replace the common exponential distribution, Lindley-type distributions have been applied to the life data analysis of several products [51]. Subsequently, several authors extended basic Lindley distribution in a variety of ways. Mazucheli and Achcar [52] assumed the Lindley distribution to analyze competing risks lifetime data. The power Lindley distribution was proposed by Ghitany et al. [53] and applied to the analysis of tensile data in carbon fibers. When the relevant two random variables obey the Lindley distribution, Al-Mutairi et al. [54] considered an estimation of the stress-strength parameters.

Nadarajah et al. [55] studied the exponentiated Lindley distribution, where the gamma, log normal and Weibull distributions are used to compare with the original Lindley distribution in term of lifetime data analysis. After that, Ashour and Eltehiwy [56] developed the exponentiated power Lindley distribution which can be regarded as the combination of the power Lindley distribution [53] and the exponentiated Lindley distribution [55]. The gamma Lindley distribution was further studied by Nedjar and Zeghdoudi [57], by applied to the failure time data of electronic components and the number of cycles to failure for specimens of yarn. In order to analyze the failure time of electronic devices and the failure stresses of carbon fibers, Ghitany et al. [58] and Mazucheli et al. [59] also proposed the weighted Lindley distribution. Recently, the Gompertz Lindley distribution was carefully examined by Baqer [60], and several candidate distributions were compared with it. We name these probability distributions developed from basic Lindley the *Lindley-type distributions*. Over the recent years, Xiao et al. [49] utilized seven Lindley-type distributions in type-I NHPP-based software reliability modeling and investigated the goodness-of-fit and predictive performances in several fault count time-interval data which were

collected in the actual software development projects.

The original Lindley distribution is defined by the c.d.f:

$$F(t) = 1 - \left(1 + \frac{at}{a+1}\right) \exp(-at), \quad t > 0, \ a > 0. \tag{3.15}$$

It is consists of an exponential distribution with scale a and a gamma distribution having shape parameter 2 and scale parameter a with the mixing proportion $a/(a+1)$ [50]. Hence, as a two-component mixture, the corresponding probability density function (p.d.f.), $f(t) = dF(t)/dt$, when $a < 1$, is shown as

$$f(t) = \frac{a^2}{a+1}(1+t)\exp(-at) \tag{3.16}$$

with $f(0) = a^2/(a+1)$ and $f(\infty) = 0$. This also shows that the p.d.f. of Lindley distribution increases in t or unimodal in t. Figure 3.9 depicts the p.d.f. of the Lindley distribution for different scale a values. Since the c.d.f. and p.d.f are given, it is easy to confirm that an increasing failure rate (IFR) of Lindley distribution is shown, where thefailure rate is given by

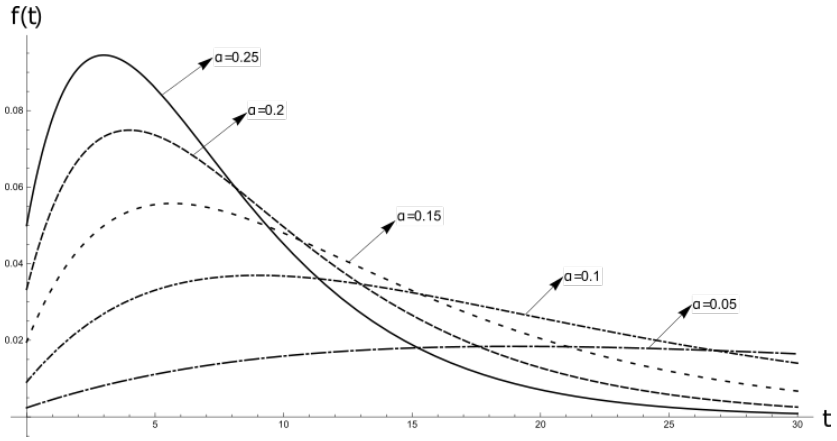


Figure 3.9: The probability density function of Lindley distribution.

$$h(t) = \frac{f(t)}{1 - F(t)} = \frac{a^2(1+t)}{a+1+at}, \tag{3.17}$$

with $h(0) = a^2/(a+1)$ and $h(\infty) = a$.

Next, to represent the fault-detection time distribution, we focus on several variations of the Lindley distribution. Set a (> 0), b (> 0) and c (> 0) as

three arbitrary parameters. Then, a total of six extensions of Lindley-type distributions are presented as follows.

(i) Gamma-Lindley Distribution [57] is a generalized Lindley distribution which is composed of the common Lindley distribution and a mixture of Gamma $(2, a)$:

$$F(t) = 1 - \frac{\{(ab + b - a)(at + 1) + a\}\exp(-at)}{b(1 + a)}. \tag{3.18}$$

The p.d.f. of Gamma-Lindley Distribution is given by

$$f(t) = \frac{a^2\{(ab + b - a)t + 1\}\exp(-at)}{b(1 + a)}. \tag{3.19}$$

(ii) Exponentiated Lindley Distribution [55] is considered as the closed form of the hazard rate with the Weibull and exponentiated exponential distributions. The c.d.f. and p.d.f. are given as:

$$F(t) = \Big\{1 - \frac{1 + a + at}{1 + a}\exp(-at)\Big\}^c, \tag{3.20}$$

$$f(t) = \frac{ca^2}{1 + a}(1 + t)\Big\{1 - \frac{1 + a + at}{1 + a}\exp(-at)\Big\}^{c-1} e^{-at}. \tag{3.21}$$

It is obvious that Equation (3.21) has two parameters, a and c, shown as a mixture of Weibull, exponentiated exponential and gamma distributions. Exponentiated Lindley distribution is reduced to the common Lindley distribution when $c = 1$.

(iii) Power Lindley Distribution [53] was considered as a power transformation $t = x^{\frac{1}{b}}$ to Equation (3.16) where the c.d.f. and p.d.f. are given as:

$$F(x) = 1 - \Big(1 + \frac{ax^b}{1 + a}\Big)\exp(-ax^b), \tag{3.22}$$

$$f(x) = \frac{ba^2}{1 + x^b}x^{b-1}\exp(-ax^b). \tag{3.23}$$

It is shown as a mixture of generalized gamma distribution with shape parameters 2 and Weibull distribution with scale parameter a and shape parameter b.

(iv) Exponentiated Power Lindley Distribution [56] is defined as a three-component mixture. it involves the common Lindley distribution, Exponentiated Lindley distribution and Power Lindley distribution. This distribution is

considered as more flexible than each component in describing different types of actual data.

$$F(t) = 1 - \left\{1 - \left(1 + \frac{at^b}{1+a}\right)\exp(-at^b)\right\}^c. \tag{3.24}$$

$$f(t) = \frac{ca^2bt^{b-1}}{a+1}(1+t^b)e^{-at^b}\left[1 - \left(1 + \frac{at^b}{a+1}\right)e^{-at^b}\right]^{c-1}. \tag{3.25}$$

(v) Gompertz Lindley Distribution [60]:

$$F(t) = 1 - \left(\frac{a^2}{1+a}\right)\frac{a + \exp(bt)}{(a - 1 + \exp(bt))^2}. \tag{3.26}$$

(vi) Weighted Lindley Distribution [58],[59]:

$$F(t) = 1 - \frac{(a+b)\Gamma_2(b, at) + (at)^b\exp(-at)}{(a+b)\Gamma_1(b)}, \tag{3.27}$$

$$f(t) = \frac{a^{b+1}t^{b-1}(1+t)\exp(-at)}{(a+b)\Gamma_1(b)}, \tag{3.28}$$

where $\Gamma_1(a) = \int_0^\infty x^{a-1}e^{-x}\mathrm{d}x$ and $\Gamma_2(a,b) = \int_b^\infty x^{a-1}e^{-x}\mathrm{d}x$. Note that when $b = 1$, the weighted Lindley distribution reduces to the Lindley distribution.

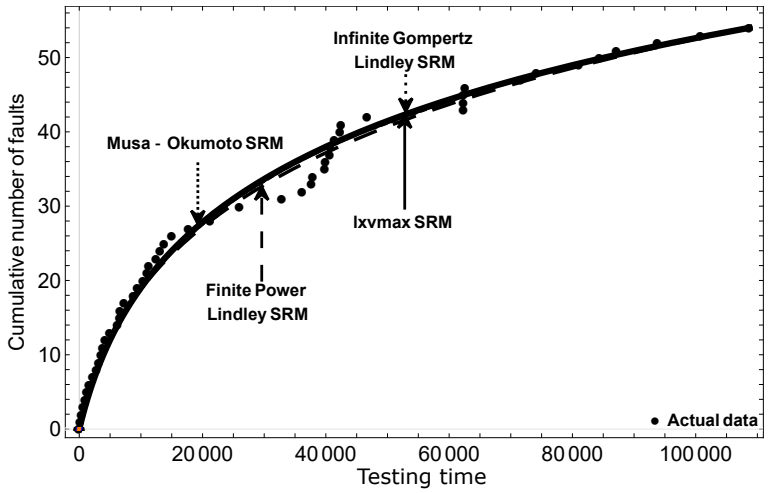
3.2.2 Type-I and Type-II Lindley-type NHPP-based SRMs

In the previous subsection, we have shown that it was possible to obtain two quite different NHPP-based SRMs; type-I NHPP-based SRM and type-II NHPP-based SRM, by importing any software fault-detection time distribution into the type-I and type-II NHPP-based software reliability modeling assumptions. Hence, we can obtain the corresponding type-I and type-II Lindley-type NHPP-based SRMs, by considering seven Lindley-type time distributions c.d.f.s shown in Equations (3.15), (3.18), (3.20), (3.22), (3.24), (3.26) and (3.27). The mean value functions of type-I and type-II Lindley-type NHPP-based SRMs are shown in Table 3.9 and Table 3.10.

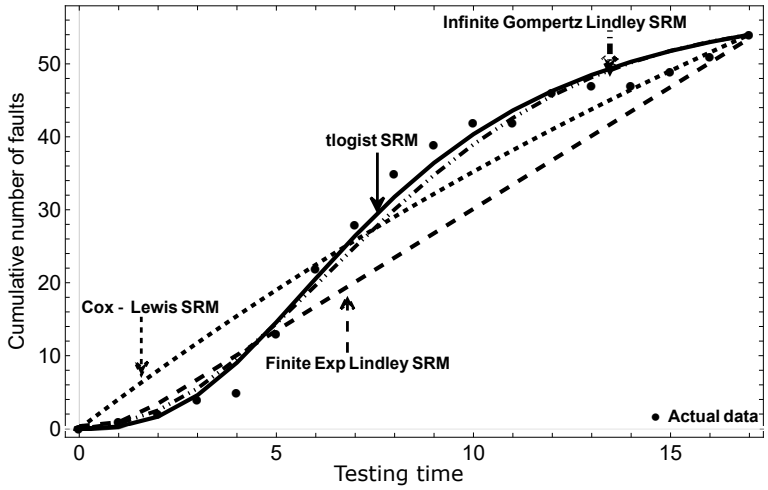
3.2.3 Numerical Experiments

3.2.3.1 Goodness-of-fit Performances

The plot for the best Lindley-type SRM and the best SRATS SRM with TDDS1 and TIDS1 are illustrated in Figure 3.10 (a) and (b), respectively. At first glance, in TDDS1, it can be seen that the four SRMs are similar in Figure 3.10 (a). It is



(a) TDDS1.



(b) TIDS1.

Figure 3.10: Cumulative number of software faults detected by type-I and type-II Lindley-type and existing NHPP-based SRMs.

difficult to identify a specific SRM with a better performance on the goodness-of-fit in this situation. But in contrast, in Figure 3.10 (b), in TDDS1, we can obviously find that two type-I NHPP-based SRMs can show more accurate and complex software fault-detection behavior. The main objectives of our numerical experiments are to derive the maximum likelihood estimates of the parameters of 11 existing type-I NHPP-based SRMs, which are showed in Table 3.1, 3 existing type-I NHPP-based SRMs (Musa-Okumoto SRM, Cox-Lewis SRM and Duane SRM), and 14 type-I and type-II Lindley-type NHPP-based SRMs considered in this section. We investigate and compare the goodness-of-fit and predictive performances of the above NHPP-based SRMs under several criteria based on the likelihood estimation in a total of 16 actual software fault count data sets.

In Table 3.11 and Table 3.12, we make a comparison of the best type-I and type-II Lindley-type SRMs with the best existing type-I and type-II NHPP-based SRMs in terms of AIC and MSE with time-domain data and group data. The best AIC/MSE in each data set is represented in bold font. From Table 3.11, it is obvious that the existing type-I NHPP-based SRMs could outperform our type-I and type-II Lindley-type SRMs in AIC and MSE in almost all cases. Only in TIDS3, and TIDS2, TIDS3, and TIDS5, the type-I Lindley-type SRMs could guarantee almost similar or better AIC and MSE than the existing type-I NHPP-based SRMs. Note that although the type-II Gompertz Lindley SRM outperformed the seven type-II Lindley-type SRMs in almost all cases, it hardly guarantee better AIC or MSE in comparison with the other three type of NHPP-based SRMs.

In Table 3.12, our Lindley-type SRMs could provide the better AIC in half of the cases (TDDS3∼TDDS6), and MSE in TDDS3, TDDS4 and TDDS5. In the group data, the result suggests that the Gompertz Lindley distribution and Lindley distribution tend to be the best fault-detection time distribution in the type-I and type-II Lindley-type NHPP-based software reliability modeling. Hence, we are optimistic to claim that Lindley-type SRMs still have a better potential ability to describe software fault count data. On one hand, we also believe that it is still of great significance to investigate the predictive performance of the Lindley-type SRMs, because, as well as we know, the goodness-of-fit performance and the predictive performance do not have an inevitable connection.

3.2.3.2 Predictive Performances

In the second experiment, we focus on the predictive performance of the Lindley-type SRMs. Figure 3.11 depicts the predicted number of detected faults after 20%, 50%, and 80% observation points in TDDS1 by the existing type-I and type-II NHPP-based SRMs, and the type-I and type-II Lindley-type SRMs with minimum PMSE. In Figure 3.11 (a), (b) and (c), the type-I Exp power Lindley-type SRM in Table 3.9, Log-extreme-value max distribution NHPP-based SRM [16], Pareto NHPP-based SRM [7] and Musa-Okumoto SRM [2, 14] provided more accurate predictive performances at 20%, 50% and 80% points in TDDS1. On one hand, we can observe that the type-II Gompertz Lindley-type SRM in Table 3.10 SRMs can not accurately predict the future trend of software debugging. But we still need a more specific investigation to evaluate the predictive performance of our Lindley-type NHPP-based SRMs in both time-domain and group data.

Table 3.5: Prediction results in time-domain data.

	NHPP-based Type-I		NHPP-based Type-II	
20% observation point				
	Best SRM	PMSE	Best SRM	PMSE
TDDS1	Lxvmax	5.073	Pareto (Musa-Okumoto)	6.420
TDDS2	Tnorm	**42.104**	Pareto (Musa-Okumoto)	145.648
TDDS3	Lxvmax	**32.131**	Power-law	1417.110
TDDS4	Lnorm	**56.477**	Pareto (Musa-Okumoto)	198.490
TDDS5	Exp	9177.670	Tlogist	**467.320**
TDDS6	Txvmin	**83.964**	Llogist	79.614
TDDS7	Lxvmax	**32.217**	Llogist	207.592
TDDS8	Lxvmax	1852.520	Lnorm	**1474.020**
50% observation point				
TDDS1	Pareto	**6.118**	Pareto (Musa-Okumoto)	6.420
TDDS2	Txvmin	**5.874**	Llogist	11.747
TDDS3	Pareto	11.712	Pareto (Musa-Okumoto)	**10.283**
TDDS4	Tlogist	**103.504**	Txvmin (Cox-Lewis)	106.282
TDDS5	Llogist	193.903	Tlogist	**77.498**
TDDS6	Lxvmax	**10.493**	Llogist	30.944
TDDS7	Exp	4480.620	Llogist	**18.425**
TDDS8	Txvmin	3569.230	Pareto (Musa-Okumoto)	**45.344**
80% observation point				
TDDS1	Lxvmax	5.772	Power-law	**3.432**
TDDS2	Lxvmax	**0.588**	Pareto (Musa-Okumoto)	0.819
TDDS3	Lxvmax	**9.419**	Power-law	19.992
TDDS4	Txvmin	**4.253**	Txvmin (Cox-Lewis)	4.258
TDDS5	Lxvmax	**21.715**	Power-law	51.677
TDDS6	Lxvmax	**2.041**	Lxvmax	3.697
TDDS7	Txvmin	6.875	Power-law	**4.291**
TDDS8	Lxvmax	57.901	Power-law	**9.268**

Table 3.6: Prediction results in group data.

	NHPP-based Type-I		**NHPP-based Type-II**	
20% observation point				
	Best SRM	PMSE	Best SRM	PMSE
TIDS1	Gamma	220.732	Power-law	**218.763**
TIDS2	Lxvmax	**29.244**	Llogist	47.377
TIDS3	Gamma	820.049	Gamma	**171.702**
TIDS4	Exp	142.854	Tlogist	**86.083**
TIDS5	Pareto	2.628	Pareto (Musa-Okumoto)	**2.625**
TIDS6	Tlogist	98.903	Llogist	**25.613**
TIDS7	Exp	387.694	Txvmin (Cox-Lewis)	**67.730**
TIDS8	Txvmin	448.935	Txvmin (Cox-Lewis)	**423.360**
50% observation point				
TIDS1	Tlogist	**96.992**	Pareto (Musa-Okumoto)	159.545
TIDS2	Txvmin	30.786	Power-law	**3.722**
TIDS3	Lxvmax	**564.782**	Gamma	849.736
TIDS4	Exp	101.303	Pareto (Musa-Okumoto)	**101.258**
TIDS5	Exp	**0.344**	Pareto (Musa-Okumoto)	0.347
TIDS6	Pareto	365.493	Gamma	**18.825**
TIDS7	Lxvmax	**22.894**	Gamma	27.045
TIDS8	Txvmin	**29.097**	Llogist	156.329
80% observation point				
TIDS1	Lnorm	**1.762**	Llogist	8.736
TIDS2	Exp	0.464	Txvmin (Cox-Lewis)	0.464
TIDS3	Tnorm	**0.331**	Txvmin (Cox-Lewis)	41.228
TIDS4	Tnorm	**1.850**	Llogist	18.985
TIDS5	Tnorm	0.224	Tlogist	**0.090**
TIDS6	Lnorm	**3.432**	Llogist	6.144
TIDS7	Txvmin	**6.118**	Llogist	17.300
TIDS8	Lxvmax	**0.864**	Llogist	6.333

Table 3.7: Software reliability assessment in time-domain data.

	Type-I		**Type-II**	
	Best SRM	**Reliability**	**Best SRM**	**Reliability**
TDDS1	Lxvmax	2.631E-06	Pareto (Musa-Okumoto)	**2.674E-06**
TDDS2	Lxvmax	**3.283E-04**	Txvmin (Cox-Lewis)	4.694E-08
TDDS3	Lxvmax	**3.687E-03**	Pareto (Musa-Okumoto)	3.751E-07
TDDS4	Lxvmax	**2.453E-04**	Pareto (Musa-Okumoto)	2.398E-04
TDDS5	Txvmin	**4.573E-01**	Txvmin (Cox-Lewis)	3.231E-03
TDDS6	Exp	**1.035E-05**	Power-law	2.596E-08
TDDS7	Pareto	**8.971E-06**	Pareto (Musa-Okumoto)	7.736E-06
TDDS8	Lxvmin	**4.592E-05**	Pareto (Musa-Okumoto)	2.516E-10

Table 3.8: Software reliability assessment in group data.

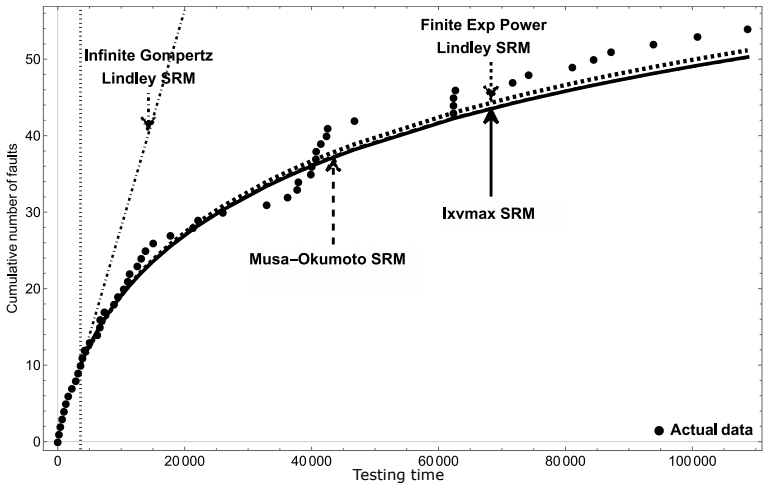
	Type-I		**Type-II**	
	Best SRM	**Reliability**	**Best SRM**	**Reliability**
TIDS1	Llogist	**4.152E-03**	Tlogist	2.217E-25
TIDS2	Lxvmax	**7.236E-05**	Llogist	6.264E-05
TIDS3	Tnorm	**3.865E-02**	Txvmin (Cox-Lewis)	2.203E-23
TIDS4	Tlogist	**2.816E-01**	Txvmin (Cox-Lewis)	3.221E-27
TIDS5	Exp	**9.832E-04**	Exp	1.234E-04
TIDS6	Lxvmax	1.939E-07	Llogist	**3.892E-07**
TIDS7	Txvmin	**9.633E-01**	Tlogist	1.280E-27
TIDS8	Llogist	**6.373E-01**	Llogist	4.052E-10

Table 3.9: Type-I Lindley-type NHPP-based SRMs.

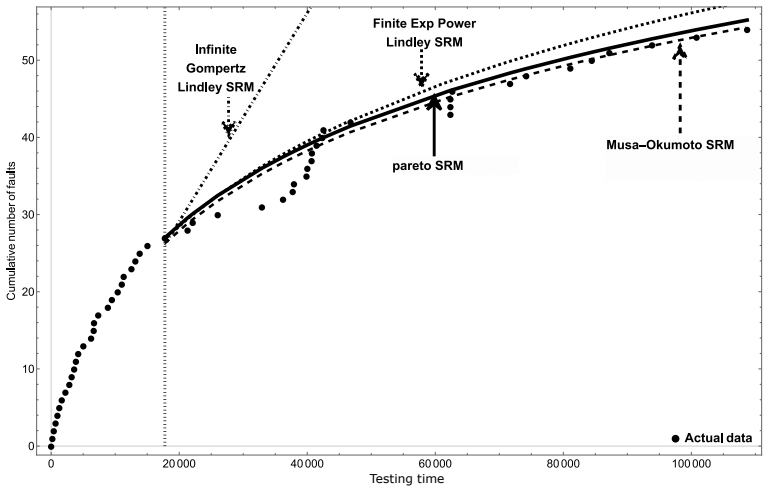
SRM	**Mean value function**
Type-I Lindley	$M(t;\boldsymbol{\theta}) = \omega\left(1 - \left(1 + \frac{at}{a+1}\right)\exp(-at)\right)$
Type-I Gamma Lindley	$M(t;\boldsymbol{\theta}) = \omega\left(1 - \frac{\{(ab+b-a)(at+1)+a\}\exp(-at)}{b(1+a)}\right)$
Type-I Exp Lindley	$M(t;\boldsymbol{\theta}) = \omega\left(\left\{1 - \frac{1+a+at}{1+a}\exp(-at)\right\}^c\right)$
Type-I Power Lindley	$M(t;\boldsymbol{\theta}) = \omega\left(1 - \left(1 + \frac{at^b}{1+a}\right)\exp(-at^b)\right)$
Type-I Exp Power Lindley	$M(t;\boldsymbol{\theta}) = \omega\left(1 - \left\{1 - \left(1 + \frac{at^b}{1+a}\right)\exp(-at^b)\right\}^c\right)$
Type-I Gompertz Lindley	$M(t;\boldsymbol{\theta}) = \omega\left(1 - \left(\frac{a^2}{1+a}\right)\frac{a+\exp(bt)}{(a-1+\exp(bt))^2}\right)$
Type-I Weighted Lindley	$M(t;\boldsymbol{\theta}) = \omega\left(1 - \frac{(a+b)\Gamma_2(b,at)+(at)^b\exp(-at)}{(a+b)\Gamma_1(b)}\right)$

Table 3.10: Type-II Lindley-type NHPP-based SRMs.

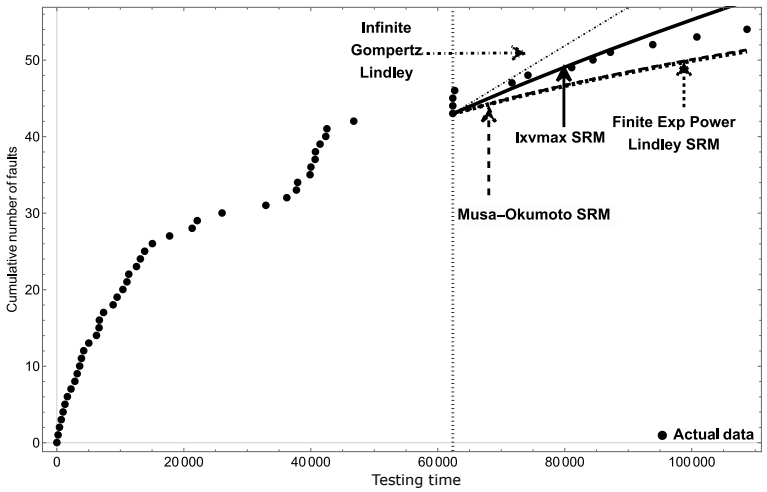
SRM	**Mean value function**
Type-II Lindley	$M(t;\boldsymbol{\alpha}) = at - \ln(at+a+1) + \ln(a+1)$
Type-II Gamma Lindley	$M(t;\boldsymbol{\alpha}) = -\ln((ab-a+b)(at+1)+a)$ $+at + \ln(a+1) + \ln(b)$
Type-II Exp Lindley	$M(t;\boldsymbol{\alpha}) = -\ln\left(1 - \left(1 - \left(\frac{at}{a+1}+1\right)e^{-at}\right)^c\right)$
Type-II Power Lindley	$M(t;\boldsymbol{\alpha}) = -\ln\left(at^b + (a+1)\right) + at + \ln(a+1)$
Type-II Exp Power Lindley	$M(t;\boldsymbol{\alpha}) = -\ln\left(1 - \left(1 - \left(\frac{at^b}{a+1}+1\right)e^{-at^b}\right)^c\right)$
Type-II Gompertz Lindley	$M(t;\boldsymbol{\alpha}) = 2\ln(a+e^{bt}-1) - \ln(a+e^{bt})$ $-2\ln(a) + \ln(a+1)$
Type-II Weighted Lindley	$M(t;\boldsymbol{\alpha}) = -\ln\left(e^{-at}(at)^b + (a+b)\Gamma(b,at)\right)$ $+\ln(a+b) + \ln(\Gamma(b))$



(a) 20% observation point.



(b) 50% observation point.



(c) 80% observation point.

Figure 3.11: Predicted the cumulative number of software faults in TDDS1.

Table 3.11: Goodness-of-fit performances based on AIC (time-domain data).

	Lindley-type NHPP-based SRMs				**Existing NHPP-based SRMs**			
	Type-I Lindley		**Type-II Lindley**		**Type-I NHPP**		**Type-II NHPP**	
	AIC	**MSE**	**AIC**	**MSE**	**AIC**	**MSE**	**AIC**	**MSE**
TDDS1	898.04 (Power Lindley)	2.86	917.57 (Gompertz Lindley)	143.47	896.67 (Lxvmax)	**1.95**	**895.31 (Musa-Okumoto)**	2.32
TDDS2	600.53 (Exp Power Lindley)	**1.68**	615.47 (Exp Power Lindley)	102.21	**598.13 (Lxvmax)**	1.71	**596.50 (Musa-Okumoto)**	1.81
TDDS3	**1938.20 (Power Lindley)**	**6.00**	2007.67 (Power Lindley)	976.81	**1938.20 (Lxvmin)**	6.57	1939.60 (Musa-Okumoto)	8.05
TDDS4	759.95 (Gompertz Lindley)	4.16	819.03 (Gompertz Lindley)	297.70	**759.58 (Txvmin)**	**3.75**	759.95 (Cox-Lewis)	5.51
TDDS5	758.53 (Exp Lindley)	**16.82**	764.60 (Gompertz Lindley)	83.58	**757.87 (Exp)**	19.99	757.03 (Duane)	19.32
TDDS6	724.45 (Power Lindley)	2.19	742.73 (Gompertz Lindley)	78.80	**721.93 (Lxvmax)**	**1.44**	726.05 (Cox-Lewis)	2.80
TDDS7	1009.40 (Power Lindley)	7.58	1027.64 (Gompertz Lindley)	92.25	**1008.22 (Lxvmax)**	**5.97**	1007.10 (Musa-Okumoto)	7.04
TDDS8	2511.64 (Exp Power Lindley)	58.34	2623.60 (Gompertz Lindley)	964.36	**2504.17 (Pareto)**	**47.40**	2503.37 (Musa-Okumoto)	63.70

Table 3.12: Goodness-of-fit performances based on AIC (group data).

	Lindley-type NHPP-based SRMs				**Existing NHPP-based SRMs**			
	Type-I Lindley		**Type-II Lindley**		**Type-I NHPP**		**Type-II NHPP**	
	AIC	**MSE**	**AIC**	**MSE**	**AIC**	**MSE**	**AIC**	**MSE**
TIDS1	74.49 (Exp Lindley)	5.52	85.34 (Gompertz Lindley)	48.55	**73.05 (Tlogist)**	**4.12**	86.94 (Cox-Lewis)	25.51
TIDS2	63.84 (Exp Power Lindley)	3.64	66.33 (Power Lindley)	32.35	**61.69 (Lxvmax)**	3.24	62.67 (Cox-Lewis)	**3.02**
TIDS3	**87.27 (Gompertz Lindley)**	6.61	107.87 (Weighted Lindley)	201.92	87.29 (Tnorm)	**6.15**	91.92 (Cox-Lewis)	31.23
TIDS4	**51.05 (Gompertz Lindley)**	**2.32**	66.65 (Lindley)	57.03	51.08 (Tlogist)	1.97	63.56 (Cox-Lewis)	27.20
TIDS5	31.22 (Weighted Lindley)	**0.13**	**27.91 (Lindley)**	0.27	29.90 (Exp)	0.12	29.91 (Cox-Lewis)	0.14
TIDS6	111.36 (Exp Power Lindley)	21.22	**104.28 (Lindley)**	22.90	108.83 (Lxvmax)	22.51	112.38 (Cox-Lewis)	**19.75**
TIDS7	126.93 (Gompertz Lindley)	3.81	138.03 (Gompertz Lindley)	24.98	**123.27 (Txvmin)**	**2.122**	141.13 (Duane)	30.71
TIDS8	120.63 (Gompertz Lindley)	10.44	150.38 (Power Lindley)	430.92	**117.47 (Llogist)**	**9.41**	174.17 (Duane)	104.51

To investigate the predictive performance, we apply the performance metrics; predictive mean squared error (PMSE). For the time-domain data, suppose that the observed software fault time sequence consists of $(t_1, t_2, ..., t_m)$, and l $(= 1, 2, \cdots)$, where l is the predictive time length and is a positive integer. In the sense of predictive performance, PMSE and MSE have the same evaluation scale; the smaller the result, the better the SRM.

The minimum PMSE in the existing NHPP-based SRMs and the Lindley-type NHPP-based SRMs for time-domain data are shown in Tables 3.13, 3.14 and 3.15, where the best SRMs are calculated by the future data (t_{m+j}, n_{m+j}) $(j = 1, 2, \cdots, l)$ obtained ex post facto. As demonstrated in Figure 3.11, the predictive performance of our type-II Lindley-type NHPP-based SRM in TDDS1 is not as good as expected. In the early and middle prediction phases, only one case in TDDS6 at 20% observation point and TDDS5 at 50% observation point shows that the Lindley-type SRM predicts the number of detected faults more accurately than the existing type-I and type-II NHPP-based SRMs. However, as the testing time goes on, except in 2 cases with TDDS5 and TDDS6, the existing NHPP-based SRMs show the smaller PMSE in almost all cases than the Lindley-type SRMs. In the later prediction phase (at 80% observation point), the type-II Lindley-type SRM is even less able to guarantee the smaller PMSE in all data sets. But on the other hand, we observed that the type-I Lindley-type SRMs could provide the smaller PMSE in three cases (TDDS1, TDDS5 and TDDS8) at 20% observation point, three cases (TDDS2, TDDS4 and TDDS) at 50% observation point, and three cases (TDDS4, TDDS5 and TDDS6) at 80% observation point.

In Tables 3.16, 3.17 and 3.18, the PMSE of the type-I and type-II Lindley-type SRMs and the existing NHPP-based SRMs are also compared when the group data are available. We can observe that our type-II Lindley-type SRMs could provide the lower PMSE in some cases; i.e., half of the data sets at 20% observation point, 3 out of 8 data sets at 50% observation point and one case at 80% observation point. On the other hand, the type-I Lindley-type SRMs outperformed the existing NHPP-based SRMs in 2 data sets at 20% observation point, 3 out of 8 data sets at 50% observation point and 3 out of 8 data sets at 80% observation point.

Hence, we can summarize that our type-II Lindley-type SRMs have better prediction accuracy at the early of software fault prediction, but that prediction accuracy continuously diminishes as the testing process proceeds. The type-II Lindley-type SRMs, on the other hand, performed more smoothly. Therefore, overall, our Lindley-type NHPP-based SRMs have the better predictive performance than the existing NHPP-based SRMs in the group data, because the Lindley-type NHPP-based SRMs could guarantee smaller PMSEs in more than half of the data sets regardless of the phase of software fault-detection prediction. Since, generally, PMSE is recognized as the most plausible prediction metric, we believe that for software fault prediction, the Lindley-type SRMs can be considered as attractive as the existing NHPP-based SRMs.

Table 3.13: Prediction results in time-domain data (20 % observation point).

	Lindley-type NHPP-based SRMs		**Existing NHPP-based SRMs**	
	Best Type-I Lindley	**Best Type-II Lindley**	**Best Type-I NHPP**	**Best Type-II NHPP**
	PMSE	**PMSE**	**PMSE**	**PMSE**
TDDS1	**4.61 (Exp Power Lindley)**	18298.74 (Gompertz Lindley)	5.07 (Lxvmax)	6.42 (Musa-Okumoto)
TDDS2	291.85 (Exp Power Lindley)	229.84 (Gompertz Lindley)	**42.104 (Tnorm)**	145.65 (Musa-Okumoto)
TDDS3	177.42 (Exp Lindley)	7128.28 (Exp Power Lindley)	**32.13 (Lxvmax)**	6.97 (Duane)
TDDS4	137.85 (Exp Power Lindley)	417.40 (Exp Power Lindley)	**56.48 (Lnorm)**	1417.11 (Duane)
TDDS5	**194.33 (Lindley)**	1348.06 (Weighted Lindley)	9177.67 (Exp)	198.49 (Musa-Okumoto)
TDDS6	259.92 (Exp Lindley)	**80.90 (Power Lindley)**	83.96 (Txvmin)	81.92 (Musa-Okumoto)
TDDS7	71.49 (Exp Power Lindley)	361.34 (Exp Power Lindley)	**32.22 (Lxvmax)**	775.85 (Musa-Okumoto)
TDDS8	**723.31 (Lindley)**	98507.90 (Gompertz Lindley)	1852.52 (Lxvmax)	27012.60 (Duane)

Table 3.14: Prediction results in time-domain data (50 % observation point).

	Lindley-type NHPP-based SRMs		**Existing NHPP-based SRMs**	
	Best Type-I Lindley	**Best Type-II Lindley**	**Best Type-I NHPP**	**Best Type-II NHPP**
	PMSE	**PMSE**	**PMSE**	**PMSE**
TDDS1	10.08 (Exp Power Lindley)	1868.97 (Gompertz Lindley)	**6.12** **(Pareto)**	6.42 (Musa-Okumoto)
TDDS2	**3.70** **(Gompertz Lindley)**	13.12 (Exp Lindley)	5.87 (Txvmin)	162.91 (Musa-Okumoto)
TDDS3	455.74 (Gamma Lindley)	9220.72 (Gompertz Lindley)	11.71 (Pareto)	**10.28** **(Musa-Okumoto)**
TDDS4	**68.14** **(Weighted Lindley)**	2612.84 (Gompertz Lindley)	103.504 (Tlogist)	106.28 (Cox-Lewis)
TDDS5	75.55 (Gamma Lindley)	**41.10** **(Lindley)**	193.90 (Llogist)	77.71 (Cox-Lewis)
TDDS6	18.82 (Lindley)	270.85 (Gamma Lindley)	**10.49** **(Lxvmax)**	73.73 (Musa-Okumoto)
TDDS7	**322.72** **(Gompertz Lindley)**	4555.59 (Gompertz Lindley)	4480.62 (Exp)	4473.65 (Musa-Okumoto)
TDDS8	146.70 (Lindley)	2.33E+05 (Power Lindley)	3569.23 (Txvmin)	**45.34** **(Musa-Okumoto)**

Table 3.15: Prediction results in time-domain data (80 % observation point).

	Lindley-type NHPP-based SRMs		**Existing NHPP-based SRMs**	
	Best Type-I Lindley	**Best Type-II Lindley**	**Best Type-I NHPP**	**Best Type-II NHPP**
	PMSE	**PMSE**	**PMSE**	**PMSE**
TDDS1	5.55 (Exp Power Lindley)	26.64 (Gompertz Lindley)	5.77 (Lxvmax)	**3.43** **(Duane)**
TDDS2	0.60 (Exp Power Lindley)	17.89 (Lindley)	**0.59** **(Lxvmax)**	0.82 (Musa-Okumoto)
TDDS3	33.83 (Power Lindley)	304.43 (Gompertz Lindley)	**9.42** **(Lxvmax)**	19.99 (Duane)
TDDS4	**3.90** **(Gamma Lindley)**	1659.57 (Gompertz Lindley)	**4.25** **(Txvmin)**	4.26 (Cox-Lewis)
TDDS5	**5.02** **(Lindley)**	101.61 (Gompertz Lindley)	21.72 (Lxvmax)	51.68 (Duane)
TDDS6	**1.59** **(Exp Power Lindley)**	118.59 (Power Lindley)	2.04 (Lxvmax)	20.25 (Musa-Okumoto)
TDDS7	13.47 (Exp Power Lindley)	15.13 (Gompertz Lindley)	6.88 (Lxvmax)	**4.29** **(Duane)**
TDDS8	119.44 (Exp Power Lindley)	59.17 (Gompertz Lindley)	57.90 (Lxvmax)	**9.27** **(Duane)**

Table 3.16: Prediction results in group data (20 % observation point).

	Lindley-type NHPP-based SRMs		**Existing NHPP-based SRMs**	
	Best Type-I Lindley	**Best Type-II Lindley**	**Best Type-I NHPP**	**Best Type-II NHPP**
	PMSE	**PMSE**	**PMSE**	**PMSE**
TIDS1	**93.00 (Lindley)**	393.22 (Weighted Lindley)	220.73 (Gamma)	218.76 (Duane)
TIDS2	36.26 (Exp Power Lindley)	735.16 (Power Lindley)	**29.24 (Lxvmax)**	988.50 (Musa-Okumoto)
TIDS3	225.24 (Weighted Lindley)	**123.01 (Gompertz Lindley)**	820.05 (Gamma)	825.88 (Duane)
TIDS4	105.71 (Gompertz Lindley)	**66.57 (Power Lindley)**	142.85 (Exp)	172.56 (Cox-Lewis)
TIDS5	2.40 (Gamma Lindley)	**0.33 (Lindley)**	2.63 (Pareto)	2.63 (Musa-Okumoto)
TIDS6	117.59 (Lindley)	1395.42 (Gompertz Lindley)	**98.90 (Tlogist)**	847.34 (Musa-Okumoto)
TIDS7	859.37 (Lindley)	**53.32 (Weighted Lindley)**	387.69 (Exp)	67.73 (Cox-Lewis)
TIDS8	**311.52 (Gamma Lindley)**	1001.72 (Gamma Lindley)	448.94 (Txvmin)	423.36 (Cox-Lewis)

Table 3.17: Prediction results in group data (50 % observation point).

	Lindley-type NHPP-based SRMs		**Existing NHPP-based SRMs**	
	Best Type-I Lindley	**Best Type-II Lindley**	**Best Type-I NHPP**	**Best Type-II NHPP**
	PMSE	**PMSE**	**PMSE**	**PMSE**
TIDS1	**20.74 (Gompertz Lindley)**	146.69 (Lindley)	96.99 (Tlogist)	159.55 (Musa-Okumoto)
TIDS2	19.72 (Exp Power Lindley)	14.80 (Power Lindley)	30.79 (Txvmin)	**3.72 (Duane)**
TIDS3	**184.01 (Lindley)**	792.88 (Weighted Lindley)	564.78 (Lxvmax)	1157.94 (Musa-Okumoto)
TIDS4	668.17 (Exp Power Lindley)	**95.31 (Gompertz Lindley)**	101.30 (Lxvmax)	101.26 (Musa-Okumoto)
TIDS5	0.29 (Gamma Lindley)	**0.14 (Power Lindley)**	0.34 (Exp)	0.35 (Musa-Okumoto)
TIDS6	354.89 (Exp Power Lindley)	**20.65 (Power Lindley)**	365.49 (Pareto)	28.16 (Cox-Lewis)
TIDS7	**17.29 (Weighted Lindley)**	24.64 (Weighted Lindley)	22.89 (Lxvmax)	64.30 (Duane)
TIDS8	125.44 (Gompertz Lindley)	299.98 (Lindley)	**29.10 (Txvmin)**	340.03 (Musa-Okumoto)

Table 3.18: Prediction results in group data (80 % observation point).

	Lindley-type NHPP-based SRMs		**Existing NHPP-based SRMs**	
	Best Type-I Lindley	**Best Type-II Lindley**	**Best Type-I NHPP**	**Best Type-II NHPP**
	PMSE	**PMSE**	**PMSE**	**PMSE**
TIDS1	3.20 (Gamma Lindley)	25.34 (Gompertz Lindley)	**1.76** **(Lnorm)**	35.91 (Musa-Okumoto)
TIDS2	**0.36** **(Gamma Lindley)**	4.41 (Power Lindley)	**0.46** **(Exp)**	0.46 (Cox-Lewis)
TIDS3	0.34 (Gompertz Lindley)	119.25 (Weighted Lindley)	**0.33** **(Tnorm)**	41.23 (Cox-Lewis)
TIDS4	**0.41** **(Exp Power Lindley)**	85.75 (Power Lindley)	1.85 (Tnorm)	112.36 (Musa-Okumoto)
TIDS5	0.09 (Exp Lindley)	**0.08** **(Gompertz Lindley)**	0.22 (Tnorm)	0.36 (Duane)
TIDS6	**1.10** **(Lindley)**	37.54 (Power Lindley)	3.43 (Lnorm)	8.98 (Musa-Okumoto)
TIDS7	30.11 (Gompertz Lindley)	39.78 (Gamma Lindley)	**6.12** **(Txvmin)**	50.80 (Musa-Okumoto)
TIDS8	2.69 (Exp Lindley)	23.39 (Gamma Lindley)	**0.86** **(Lxvmax)**	35.28 (Musa-Okumoto)

3.2.3.3 Software Reliability Assessment

Through the experiments of goodness-of-fit and predictive performances, we are not asserting that our Lindley-type SRMs could outperform the existing NHPP-based SRMs, but, it should be emphasised that, in the NHPP-based software reliability modeling, except for the existing SRMs [43], the Lindley-type SRMs could also be good candidates. In describing the software fault-detection time distribution, the Lindley-type distributions should be good choices.

Hence, we also concern quantifying the software reliability by our Lindley-type SRMs. We define $R(x)$ as the probability that software dose contain no faults detected during a time interval $(t, t + x]$, when the software test is stopped at time t where x is the software operational time. We set as 1 time of each testing length in CPU time unit for the time-domain data or calendar time (week) for the group data.

Tables 3.19 and 3.20 present the quantitative software reliability with the time-domain and group data sets, respectively. We utilize the type-I and type-II Lindley-type SRMs and the existing NHPP-based SRMs that could provide the best AIC in the time period $(0, t)$. We indicate the software reliability value that is more close to 1 with the bold font. It can be seen that the type-I Lindley-type SRMs could provide larger software reliability than the existing NHPP-based SRMs in 5 of 8 time-domain data sets and half of the group data sets. On the other hand, Our type-II Lindley-type SRMs could not show the larger reliability in all the cases. Hence, based on these results, we can't fully claim that the Lindley-type NHPP-based SRMs can provide more optimistic quantification in software reliability assessment. However, the software reliability based on all the NHPP-based SRMs suggests that none of software projects in our experiments can be recommended for immediate market release, because, the reliability estimates are close to zero.

Table 3.19: Reliability by the best AIC SRMs (time-domain data).

	Lindley-type NHPP-based SRMs		**Existing NHPP-based SRMs**	
	Best Type-I Lindley	**Best Type-II Lindley**	**Best Type-I NHPP**	**Best Type-II NHPP**
	Reliability	**Reliability**	**Reliability**	**Reliability**
TDDS1	**8.49E-05 Power Lindley**	2.35E-19 (Lindley)	2.67E-06 (Lxvmax)	9.32E-06 (Musa-Okumoto)
TDDS2	**4.12E-03 (Exp Power Lindley)**	1.40E-09 (Exp Power Lindley)	3.75E-03 (Lxvmax)	2.40E-03 (Musa-Okumoto)
TDDS3	6.95E-12 (Power Lindley)	8.63E-60 (Power Lindley)	2.52E-10 (Lxvmin)	**2.64E-10 (Musa-Okumoto)**
TDDS4	3.53E-01 (Gompertz Lindley)	2.64E-13 (Gompertz Lindley)	**1.00E+00 (Txvmin)**	**1.00E+00 (Musa-Okumoto)**
TDDS5	8.36E-12 (Exp Lindley)	3.12E-75 (Gompertz Lindley)	**2.60E-08 (Exp)**	1.01E-19 (Duane)
TDDS6	**2.38E-01 (Power Lindley)**	3.49E-06 (Gompertz Lindley)	4.69E-03 (Lxvmax)	3.13E-17 (Musa-Okumoto)
TDDS7	**9.42E-03 (Power Lindley)**	5.82E-05 (Gompertz Lindley)	2.40E-04 (Lxvmax)	3.83E-15 (Musa-Okumoto)
TDDS8	**1.15E-05 (Exp Power Lindley)**	1.75E-07 (Gompertz Lindley)	7.74E-06 (Pareto)	1.88E-32 (Musa-Okumoto)

Table 3.20: Reliability by the best AIC SRMs (group data).

	Lindley-type NHPP-based SRMs		**Existing NHPP-based SRMs**	
	Best Type-I Lindley	**Best Type-II Lindley**	**Best Type-I NHPP**	**Best Type-II NHPP**
	Reliability	**Reliability**	**Reliability**	**Reliability**
TIDS1	1.57E-04 Exp Lindley	2.21E-25 (Gompertz Lindley)	**4.15E-03** **(Llogist)**	3.57E-24 (Cox-Lewis)
TIDS2	**7.45E-05** **(Exp Power Lindley)**	9.19E-17 (Power Lindley)	7.24E-05 (Lxvmax)	3.17E-17 (Cox-Lewis)
TIDS3	2.59E-03 (Gompertz Lindley)	1.68E-49 (Weighted Lindley)	**3.87E-02** **(Tnorm)**	6.59E-101 (Cox-Lewis)
TIDS4	**2.82E-01** **(Gompertz Lindley)**	4.61E-28 (Lindley)	2.81E-01 (Tlogist)	**9.99E-73** **(Cox-Lewis)**
TIDS5	**1.52E-02** **(Weighted Lindley)**	1.30E-28 (Lindley)	9.83E-04 (Exp)	3.01E-05 (Cox-Lewis)
TIDS6	1.38E-07 (Exp Power Lindley)	1.68E-33 (Lindley)	**1.94E-07** **(Lxvmax)**	3.33E-77 (Cox-Lewis)
TIDS7	1.31E-01 (Gompertz Lindley)	1.26E-11 (Gompertz Lindley)	**9.63E-01** **(Txvmin)**	5.47E-05 (Duane)
TIDS8	**9.66E-01** **(Gompertz Lindley)**	3.37E-17 (Power Lindley)	6.37E-01 (Llogist)	3.32E-02 (Duane)

Hence, we can conclude that in this section, we have proposed 7 novel type-II Lindley-type NHPP-based SRMs over the type-II software reliability modeling assumption. We have investigated their goodness-of-fit and predictive performances, and made comparisons under the type-I Lindley-type NHPP-based SRMs, 11 existing type-I and 3 existing NHPP-based SRMs. In most of the time-domain and group data sets, the Lindley-type NHPP-based SRMs were difficult to provide the smaller AIC when their parameters were given by the maximum likelihood estimation. However, in the group data set, the Lindley-type NHPP-based SRMs demonstrated nice predictive performances that outperform the existing NHPP-based SRMs at any phase of software testing.

3.3 Burr-type NHPP-based Software Reliability Modeling

3.3.1 Burr-Type Distributions

For a continuous random variable X with the support $(-\infty, +\infty)$, let $F(x;\boldsymbol{\alpha})$ and $f(x;\boldsymbol{\alpha})$ be the c.d.f. and the probability density function (p.d.f.), respectively, where $F(x;\boldsymbol{\alpha})$ is an absolutely continuous non-decreasing function from $F(-\infty;\boldsymbol{\alpha}) = 0$ to $F(\infty;\boldsymbol{\alpha}) = 1$. For arbitrary a and b $(a < b)$, $\Pr\{a \leq X \leq b\} = F(b;\boldsymbol{\alpha}) - F(a;\boldsymbol{\alpha}) = \int_a^b f(x;\boldsymbol{\alpha})dx$ with $F(x;\boldsymbol{\alpha}) = \int_{-\infty}^{x} f(x;\boldsymbol{\alpha})dx$ and $f(x;\boldsymbol{\alpha}) = dF(x;\boldsymbol{\alpha})/dx$. Burr [61] introduced a new family of c.d.f.s which satisfy the following differential equation;

$$\frac{dF(x;\boldsymbol{\alpha})}{dx} = F(x;\boldsymbol{\alpha})(1 - F(x;\boldsymbol{\alpha}))g(x, F(x;\boldsymbol{\alpha})), \tag{3.29}$$

where $g(x, F(x;\boldsymbol{\alpha}))$ is an arbitrary positive function with $0 \leq F(x;\boldsymbol{\alpha}) \leq 1$. If $g(x, F(x;\boldsymbol{\alpha})) = \left(b_1 + b_2 x + b_3 x^2\right)^{-1}$ and if $F(x;\boldsymbol{\alpha})$ and $1 - F(x;\boldsymbol{\alpha})$ are replaced by $f(x)$ and $(b_0 - x)$, respectively, with arbitrary constants b_0, b_1, b_2, and b_3, then Equation (3.29) is reduced to the differential equation for the well-known Pearson system;

$$\frac{df(x;\boldsymbol{\alpha})}{dx} = \frac{f(x;\boldsymbol{\alpha})(b_0 - x)}{(b_1 + b_2 x + b_3 x^2)}, \tag{3.30}$$

which leads to many popular c.d.f.s, such as Pearson-type I (beta distribution), Pearson-type III (gamma distribution), Pearson-type VIII (power distribution), Pearson-type X (exponential distribution) and Pearson-type XI (a particular class of Pareto distribution).

Burr [61] considered a special case of $g(x, F(x;\boldsymbol{\alpha})) = g(x;\boldsymbol{\alpha})$. By solving Equation (3.29), we obtain

$$F(x;\boldsymbol{\alpha}) = \frac{1}{\left[e^{-\int g(x;\boldsymbol{\alpha})dx} + 1\right]}. \tag{3.31}$$

It should be noted that the selection of the function $g(x;\boldsymbol{\alpha})$ makes the c.d.f. $F(x;\boldsymbol{\alpha})$ increase monotonously from 0 to 1 within a specified time x. The above statement is often called the *Burr hypothesis*. Finally, Burr [61] derived 12 Burr-type distributions I∼XII by considering 12 kinds of $g(x;\boldsymbol{\alpha})$ functions. Table 3.21 lists the Burr-type distributions proposed in [61].

Table 3.21: Burr-type distributions.

Type	**c.d.f.**	**Domain of x**
I	$F(x;\boldsymbol{\alpha}) = x$	$(0, 1)$
II	$F(x;\boldsymbol{\alpha}) = (e^{-x} + 1)^{-b}$	$(-\infty, +\infty)$
III	$F(x;\boldsymbol{\alpha}) = (1 + (x)^{-a})^{-b}$	$(0, +\infty)$
IV	$F(x;\boldsymbol{\alpha}) = \left(((c-x)/x)^{1/c} + 1\right)^{-b}$	$(0, c)$
V	$F(x;\boldsymbol{\alpha}) = (ae^{-\tan x} + 1)^{-b}$	$(-\pi/2, \pi/2)$
VI	$F(x;\boldsymbol{\alpha}) = \left(ae^{-\mathrm{csinh}(x)} + 1\right)^{-b}$	$(-\infty, +\infty)$
VII	$F(x;\boldsymbol{\alpha}) = 2^{-b}\left(1 + \tanh(x)\right)^{b}$	$(-\infty, +\infty)$
VIII	$F(x;\boldsymbol{\alpha}) = (\arctan(e^{x})2/\pi)^{b}$	$(-\infty, +\infty)$
IX	$F(x;\boldsymbol{\alpha}) = 1 - 2\left(a\left((1 + e^{x})^{b} - 1\right) + 2\right)^{-1}$	$(-\infty, +\infty)$
X	$F(x;\boldsymbol{\alpha}) = \left(1 - e^{-(x)^2}\right)^{b}$	$(0, +\infty)$
XI	$F(x;\boldsymbol{\alpha}) = (x - (1/2\pi)\sin 2\pi x)^{b}$	$(0, 1)$
XII	$F(x;\boldsymbol{\alpha}) = 1 - (1 + x^{a})^{-b}$	$(0, +\infty)$

$(\omega{>}0, a{>}0, b{>}0, c{>}0)$

3.3.2 Type-I Burr-Type NHPP-based SRMs

The Burr-type III, X, and XII distributions were applied to describe the software fault-detection time distribution in the past literature, where these c.d.f.s have positive support $(0, \infty)$. In other words, from Table 3.21, it is immediate to see that the Burr-type I, IV, V, and XI distributions are not appropriate in modeling the software fault-detection time. In addition to the Burr-type III distribution [74, 75, 76, 77], the Burr-type X distribution [79], the Burr-type

XII distribution [64, 66, 67, 68, 69, 70, 71, 72, 73] with the positive support $x \in (0, \infty)$, it is possible to transform the c.d.f. with support $(-\infty, +\infty)$ to the log Burr-type distributions and the truncated Burr-type distributions with the support $x \in (0, \infty)$ by taking $\exp(x)$ and truncating x at the origin, respectively. So, we consider the log Burr-type II, VI, VII, VIII, IX distributions and the truncated Burr-type II, VI, VII, VIII, IX distributions to represent the mean value function of the NHPP-based SRM by

$$M(t; \boldsymbol{\theta}) = \omega F(\ln t; \boldsymbol{\alpha}), \tag{3.32}$$

and

$$M(t; \boldsymbol{\theta}) = \omega \frac{F(t; \boldsymbol{\alpha}) - F(0; \boldsymbol{\alpha})}{1 - F(0; \boldsymbol{\alpha})}, \tag{3.33}$$

respectively. The underlying idea of the log Burr-type distribution comes from the log-normal NHPP-based SRM [8, 17] and the log-logistic NHPP-based SRM [13]. In fact, it is known that the logarithmic Burr-type II distribution is reduced to the log-logistic distribution [62]. The truncation at the origin for the Burr II, VI, VII, VIII, IX distributions with the support $(-\infty, +\infty)$ is inspired by the truncated normal NHPP-based SRM [17] and the truncated logistic NHPP-based SRM [15]. Table 2.2 presents the type-I Burr-type NHPP-based SRMs considered in this section, where we applied generalized Burr-type III, VI, VII, VIII, IX, X, and XII distributions by introducing an additional scale parameter d. That is to say, if $d = 1$, then the Burr-type distributions in Table 3.22 become the original form in Table 3.21.

3.3.3 Type-II Burr-Type NHPP-based SRMs

In the previous subsection, we have specifically introduced the type-I NHPP-based SRM with Burr-type distributions. Hence, by substituting the underlying Burr-type III, VI, VII, VIII, IX, X, and XII software fault-detection time c.d.f.s (in Table 3.21) into Equation (3.5), we can derive 11 novel Burr-type NHPP-based SRMs, say, *type-II Burr-type NHPP-based SRMs*. Note that the mean value function of the type-II log Burr-type SRMs and the type-II truncated Burr-type SRMs should be modified from Equations (3.32) and (3.33) to

$$M(t; \boldsymbol{\alpha}) = -\ln(1 - F(\ln t; \boldsymbol{\alpha})) \tag{3.34}$$

$$M(t; \boldsymbol{\alpha}) = -\ln\left(1 - \frac{F(t; \boldsymbol{\alpha}) - F(0; \boldsymbol{\alpha})}{1 - F(0; \boldsymbol{\alpha})}\right), \tag{3.35}$$

Table 3.22: Type-I Burr-type NHPP-based SRMs.

Models	**Mean value function**
Burr-type III	$M(t;\boldsymbol{\theta}) = \omega\left(1+(t/d)^{-a}\right)^{-b}$
Log Burr-type VI	$M(\ln t;\boldsymbol{\theta}) = \omega\left(ae^{-c\sinh(\ln t/d)}+1\right)^{-b}$
Truncated (Tru) Burr-type VI	$M(t;\boldsymbol{\theta}) = \omega\left(\frac{\left(ae^{-c\sinh(t/d)}+1\right)^{-b}-\left(ae^{-c\sinh(0)}+1\right)^{-b}}{1-\left(ae^{-c\sinh(0)}+1\right)^{-b}}\right)$
Log Burr-type VII	$M(\ln t;\boldsymbol{\theta}) = \omega 2^{-b}\left(1+\tanh\left(\ln t/d\right)\right)^{b}$
Truncated (Tru) Burr-type VII	$M(t;\boldsymbol{\theta}) = \omega\left(\frac{2^{-b}(1+\tanh(t/d))^{b}-2^{-b}(1+\tanh(0))^{b}}{1-2^{-b}(1+\tanh(0))^{b}}\right)$
Log Burr-type VIII	$M(\ln t;\boldsymbol{\theta}) = \omega\left(\arctan(e^{\ln t/d})2/\pi\right)^{b}$
Truncated (Tru) Burr-type VIII	$M(t;\boldsymbol{\theta}) = \omega\left(\frac{\left(\arctan(e^{t/d})2/\pi\right)^{b}-(2/\pi\arctan(1))^{b}}{1-(2/\pi\arctan(1))^{b}}\right)$
Log Burr-type IX	$M(\ln t;\boldsymbol{\theta}) = \omega\left(1-2\left(a\left(\left(1+e^{\ln t/d}\right)^{b}-1\right)+2\right)^{-1}\right)$
Truncated (Tru) Burr-type IX	$M(t;\boldsymbol{\theta}) = \omega\left(\frac{\left(2^{b}+1\right)^{-1}-\left(\left(1+e^{t/d}\right)^{b}+1\right)^{-1}}{\left(2^{b}+1\right)^{-1}}\right)$
Burr-type X	$M(t;\boldsymbol{\theta}) = \omega\left(1-e^{-(t/d)^{2}}\right)^{b}$
Burr-type XII	$M(t;\boldsymbol{\theta}) = \omega\left(1-\left(\frac{1}{1+(t/d)^{a}}\right)^{b}\right)$
	$(\omega>0, a>0, b>0, c>0, d>0)$

respectively. We also utilize the generalized Burr-type III, VI, VII, VIII, IX, X and XII distributions with the introduction of an additional scale parameter d. The details of these SRMs are shown in Table 3.23.

Table 3.23: Type-II Burr-type NHPP-based SRMs.

Models	**Mean value function**
Burr-type III	$M(t;\boldsymbol{\alpha}) = -\ln\left(1-\left(\left(\frac{t}{d}\right)^{-b}+1\right)^{-c}\right)$
Log Burr-type VI	$M(\ln t;\boldsymbol{\alpha}) = -\ln\left(1-\left(b\ e^{-d\sinh\left(\frac{\ln(t)}{f}\right)}+1\right)^{-c}\right)$
Truncated (Tru) Burr-type VI	$M(t;\boldsymbol{\alpha}) = -\ln\left(\frac{(b+1)^c\left(1-\left(be^{-d\sinh\left(\frac{t}{f}\right)}+1\right)^{-c}\right)}{(b+1)^c-1}\right)$
Log Burr-type VII	$M(\ln t;\boldsymbol{\alpha}) = -\ln\left(1-2^{-b}\left(\tanh\left(\frac{\ln(t)}{c}\right)+1\right)^{b}\right)$
Truncated (Tru) Burr-type VII	$M(t;\boldsymbol{\alpha}) = -\ln\left(\frac{2^b-\left(\tanh\left(\frac{t}{c}\right)+1\right)^b}{2^b-1}\right)$
Log Burr-type VIII	$M(\ln t;\boldsymbol{\alpha}) = -\ln\left(1-\left(\frac{2\tan^{-1}\left(\exp\left(\frac{\ln(t)}{c}\right)\right)}{\pi}\right)^{b}\right)$
Truncated (Tru) Burr-type VIII	$M(t;\boldsymbol{\alpha}) = -\ln\left(\frac{2^b-\left(\frac{4}{\pi}\right)^b\tan^{-1}\left(e^{t/c}\right)^b}{2^b-1}\right)$
Log Burr-type IX	$M(\ln t;\boldsymbol{\alpha}) = -\ln\left(\frac{2}{b\left(\left(\exp\left(\frac{\ln(t)}{d}\right)+1\right)^c-1\right)+2}\right)$
Truncated (Tru) Burr-type IX	$M(t;\boldsymbol{\alpha}) = -\ln\left(\frac{b(2^c-1)+2}{b\left(\left(e^{t/d}+1\right)^c-1\right)+2}\right)$
Burr-type X	$M(t;\boldsymbol{\alpha}) = -\ln\left(1-\left(1-\exp\left(-\left(\frac{t}{c}\right)^2\right)\right)^{b}\right)$
Burr-type XII	$M(t;\boldsymbol{\alpha}) = -\ln\left(\left(\frac{1}{\left(\frac{t}{d}\right)^b+1}\right)^{c}\right)$

$(a>0, b>0, c>0, d>0)$

3.3.4 Numerical Experiments

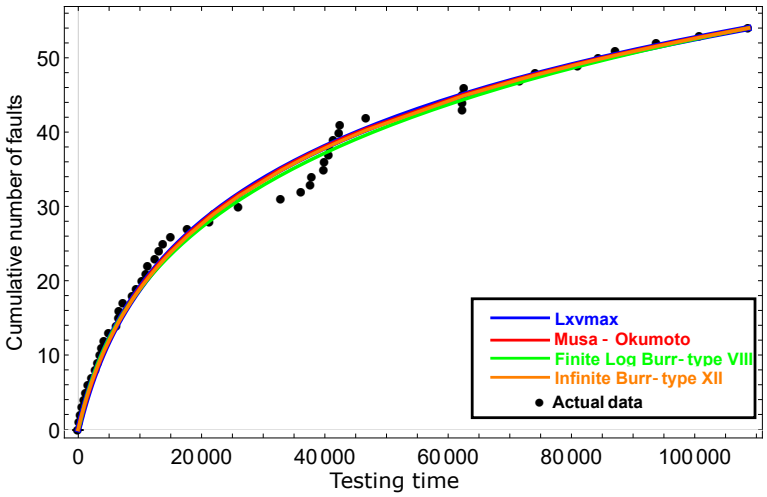
3.3.4.1 Goodness-of-fit Performances

We also utilize the maximum likelihood estimation for the parameter estimation of our type-I and type-II Burr-type NHPP-based SRMs. By maximizing $\ln\mathcal{L}(\boldsymbol{\theta}$ or $\boldsymbol{\alpha};\boldsymbol{D})$ with respect to $\boldsymbol{\theta}$ or $\boldsymbol{\alpha}$, ML estimate $\hat{\boldsymbol{\theta}}$ or $\hat{\boldsymbol{\alpha}}$ can be obtained.

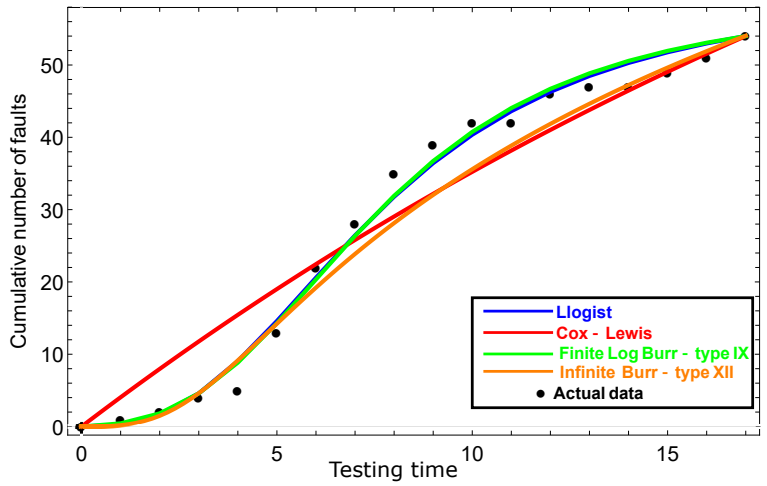
In numerical experiments, we analyze a total of 8 time-domain data sets (DS1 $\sim$ DS8 in Table 2.1 (i)), labeled TDDS1$\sim$TDDS8, and 8 group data sets

(DS15 $\sim$ DS21 in Table 2.1 (ii)), called TIDS1$\sim$TIDS8.

We compare the goodness-of-fit performances of our type-I Burr-type SRMs, and type-II Burr-type SRMs with 3 existing type-II NHPP-based SRMs (Cox-Lewis, Duane and Musa-Okumoto SRMs) and 11 existing type-I NHPP-based SRMs in [43] (see Table 3.1). We apply AIC and MSE to investigate the goodness-of-fit performances of our type-I and type-II Burr-type NHPP-based SRMs. In Figure 3.12 (a) and (b), we depict the plot for the best existing



(a) TDDS1.



(b) TIDS1.

Figure 3.12: Behavior of cumulative number of software faults with the best type-II Burr-type and the best existing type-II NHPP-based SRMs.

type-I NHPP-based SRM (blue curve), the best existing type-II NHPP-based

SRM (red curve), the best type-I Burr-type SRM (green curve), and the best type-II Burr-type SRM (orange curve) in TDDS1 and TIDS1. At first glance, in Figure 3.12 (a), 4 SRMs exhibit almost similar behavior. However, in Figure 3.12 (b), the best existing type-I NHPP and type-I Burr-type SRMs show strong abilities to fit the underlying fault count data. More specifically, in Table 3.24 and Table 3.25, we compare the best SRMs of our type-I and type-II Burr-type NHPP-based SRMs with the other two type of best SRMs; existing type-I and type-II NHPPs in time-domain data and group data, respectively. The bold font and underline indicate the SRMs with the minimum AIC and MSE in each data set. It's worth noting that the significant difference in terms of AIC might be regarded as more than 2 according to the definition of AIC. From Table 3.24, it can be seen that our type-II Burr-type SRMs could guarantee the smaller AIC than the other three type of existing NHPP-based SRMs in five cases (TDDS2, TDDS4, TDDS5, TDDS7, TDDS8), but the difference was significant in only TDDS4 and TDDS8. We also noted that the best type-II Burr-type that can guarantee the smallest AIC or MSE are all given by the truncated Burr-type SRMs. Although the AICs and MSEs show that our SRMs still cannot be fully replaced by the existing NHPP-based SRMs, our Burr-type NHPP-based SRMs should be a better candidate for selecting the best SRM in terms of goodness-of-fit.

In the group data sets (see Table 3.25), in TIDS2, TIDS5 and TIDS6, our type-II Burr-type SRMs (Log burr-type VIII and Burr-type XII) could provide the smaller AIC, but could not beat the other three type of SRMs in terms of MSE. Only TIDS1 showed a significant difference in AIC between our SRM and the other existing SRMs. On the other hand, we also notice that the type-I Burr-type SRMs provided the smallest AIC in four cases (TIDS1, TIDS3, TIDS4 and TIDS8) and the smallest MSE in 5 cases (TIDS1, TIDS3, TIDS4, TIDS5 and TIDS8). The type-I and type-II Burr-type NHPP-based SRMs completely outperform the existing type-I and type-II NHPP-based SRMs in terms of goodness-of-fit performance in group data. Even though the smallest AIC and MSE are still given by the existing NHPP-based SRM (Lxvmax) in TIDS7, but a comparison with the best type-I Burr-type in the same data set shows that the differences are quite insignificant.

Table 3.24: Goodness-of-fit performances based on AIC (time-domain data).

Data	Existing type-I NHPPs			Existing type-II NHPPs			Type-I Burr-type SRMs			Type-II Burr-type SRMs		
Set	**Best SRM**	**AIC**	**MSE**	**Best SRM**	**AIC**	**MSE**	**Best SRM**	**AIC**	**MSE**	**Best SRM**	**AIC**	**MSE**
TDDS1	Lxvmax	896.666	1.950	Musa-Okumoto	**895.305**	2.315	Log Bur-typr VIII	896.663	**1.941**	Burr-type XII	897.294	2.242
TDDS2	Lxvmax	598.131	1.705	Musa-Okumoto	596.501	1.809	Log Bur-type VIII	598.122	1.674	Tru Burr-type VIII	**595.295**	**1.557**
TDDS3	Lxvmin	**1938.160**	**6.570**	Musa-Okumoto	1939.600	8.052	Burr-type X	1939.258	13.321	Burr-type XII	1938.650	7.138
TDDS4	Txvmin	759.579	**3.747**	Cox-Lewis	759.948	5.509	Tru Burr-type IX	759.454	4.185	Tru Burr-type VII	**754.822**	3.975
TDDS5	Exp	757.869	19.985	Duane	757.031	19.315	Burr-type X	757.119	**16.910**	Tru Burr-type VI	**755.269**	22.792
TDDS6	Lxvmax	**721.928**	**1.442**	Cox-Lewis	726.052	2.803	Log Burr-type VIII	722.397	2.095	Burr-type XII	722.747	2.520
TDDS7	Lxvmax	1008.220	5.970	Musa-Okumoto	1007.100	7.039	Log Burr-type VII	1008.200	5.967	Tru Burr-type VI	**1005.730**	**4.212**
TDDS8	Pareto	2504.170	47.404	Musa-Okumoto	2503.370	63.699	Tru Burr-type VI	2502.160	50.351	Tru Burr-type VII	**2475.520**	**13.034**

Table 3.25: Goodness-of-fit performances based on AIC (time-domain data).

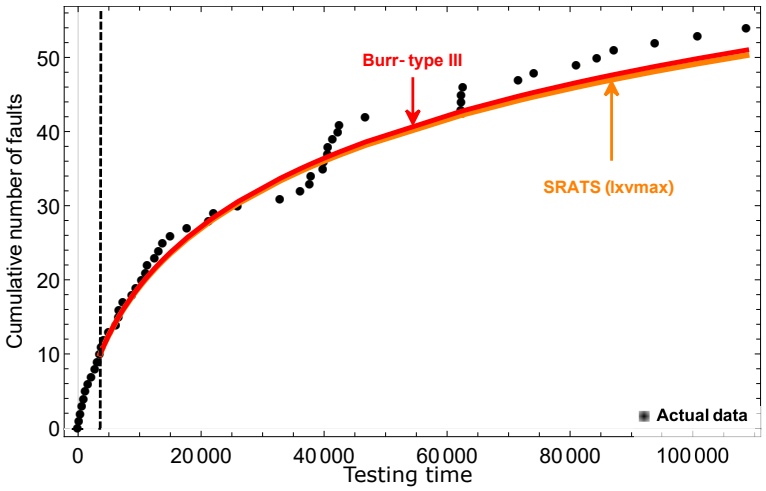
Data	**Existing type-I NHPPs**			**Existing type-II NHPPs**			**Type-I Burr-type SRMs**			**Type-II Burr-type SRMs**		
Set	**Best SRM**	**AIC**	**MSE**	**Best SRM**	**AIC**	**MSE**	**Best SRM**	**AIC**	**MSE**	**Best SRM**	**AIC**	**MSE**
TIDS1	Llogist	73.053	4.115	Cox-Lewis	86.936	25.513	Log Burr-type IX	**72.500**	**3.819**	Burr-type XII	76.115	12.840
TIDS2	Lxvmax	61.694	3.239	Cox-Lewis	62.969	**3.017**	Log Burr-type IX	59.459	3.212	Log Burr-type VIII	**54.632**	3.283
TIDS3	Tnorm	87.275	6.151	Cox-Lewis	91.919	31.232	Log Burr-type VI	**85.873**	**3.086**	Tru Burr-type VI	89.213	8.368
TIDS4	Tlogist	51.052	1.968	Cox-Lewis	63.556	27.199	Tru Burr-type IX	**50.256**	**1.816**	Tru Burr-type VI	53.217	4.117
TIDS5	Exp	29.911	0.118	Cox-Lewis	29.910	0.138	Log Burr-type IX	30.231	**0.071**	Log Burr-type VIII	**29.132**	0.567
TIDS6	Lxvmax	108.831	22.514	Cox-Lewis	112.384	**19.752**	Log Burr-type IX	103.456	20.890	Burr-type XII	**102.871**	34.921
TIDS7	Txvmin	**123.256**	**2.122**	Duane	141.128	30.708	Burr-type III	124.767	2.180	Tru Burr-type VI	129.212	2.847
TIDS8	Llogist	117.470	9.408	Duane	174.167	104.510	Log Burr-type IX	**117.234**	**9.042**	Burr-type XII	143.606	59.840

3.3.4.2 Predictive Performances

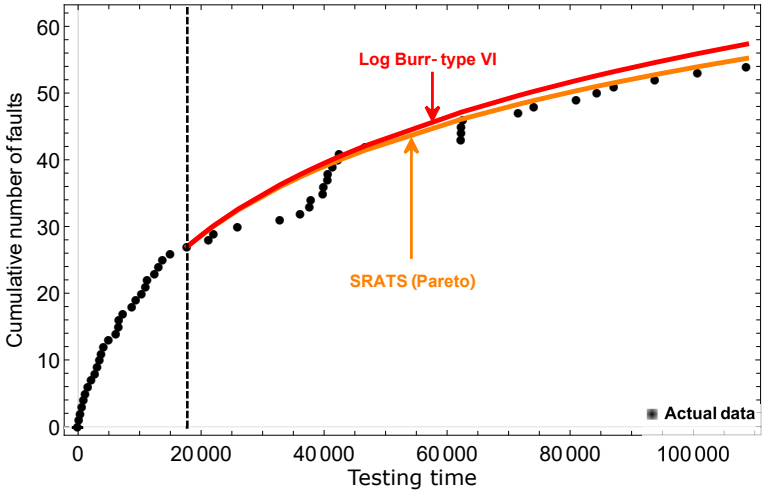
Here, we investigate the predictive performance of our type-I and type-II Burr-type NHPP-based SRMs. The predictive performance is also measured by the PMSE to evaluate the average squared distance between the predicted cumulative number of software faults and its (unknown) realization per prediction length. Our experiments set three observation points; 20%, 50%, and 80% of the whole data set, predict the cumulative number of software faults for the remaining period, say, 80%, 50%, and 20% lengths, and calculate the PMSEs in all the cases with all SRMs. The prediction length becomes shorter as the observation point is larger.

In Figures 3.13 and 3.14, we show examples of predictive behavior of the cumulative number of software faults with the Burr-type and existing SRMs in TDDS1 and TIDS1, respectively, where the dotted line denotes the prediction point. In these figures, we plot the best predictive SRMs with the minimum PMSE. In Figure 3.13, since the underlying fault-detection time behaves like an exponential curve, both SRMs; the Burr-type NHPP-based SRM and the existing NHPP-based SRM, could show a similar prediction trend. On the other hand, the group data in Figure 3.14 represents the S-shaped curve, and both SRMs resulted in the miss-prediction in the early testing phases. The trend change in the future causes these poor predictive performances. More specifically, in Figure 3.14 (a), both SRMs could not predict the S-shaped increasing trend. In Figure 3.14 (b), they failed to predict the 3 steps-increasing trends. From these results, we can understand that the prediction of the future unknown trend change is essentially difficult, even though the prediction length is relatively short.

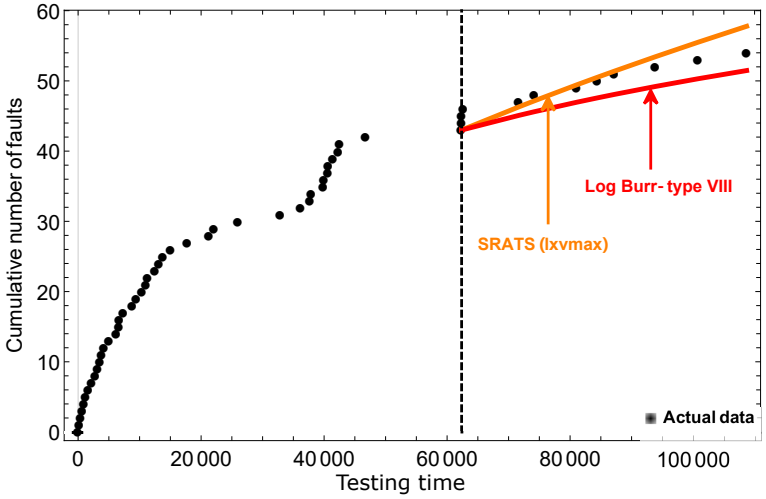
Tables 3.26 and 3.27 present the comparison results on the PMSE in time-domain data sets and group data sets, respectively, where we select the best SRM with the smallest PMSE from both the type-I and type-II Burr-type NHPP-based SRMs, and the existing NHPP-based SRMs. For the time-domain data in Table 3.26, it is seen that except in the 20% observation, the type-I and type-II Burr-type NHPP-based SRMs could guarantee the smaller PMSE than the existing NHPP-based SRMs in almost cases. When the testing phase is early (20%), the Burr-type NHPP-based SRMs also provided the smaller PMSE in 5



(a) 20% observation point.

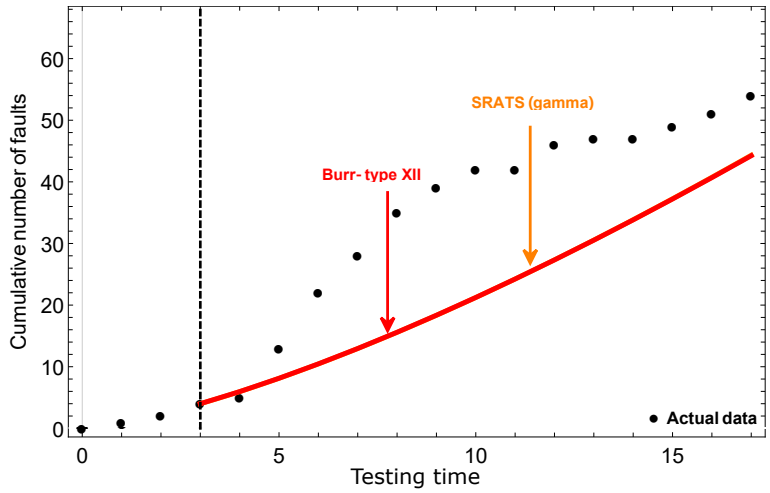


(b) 50% observation point.

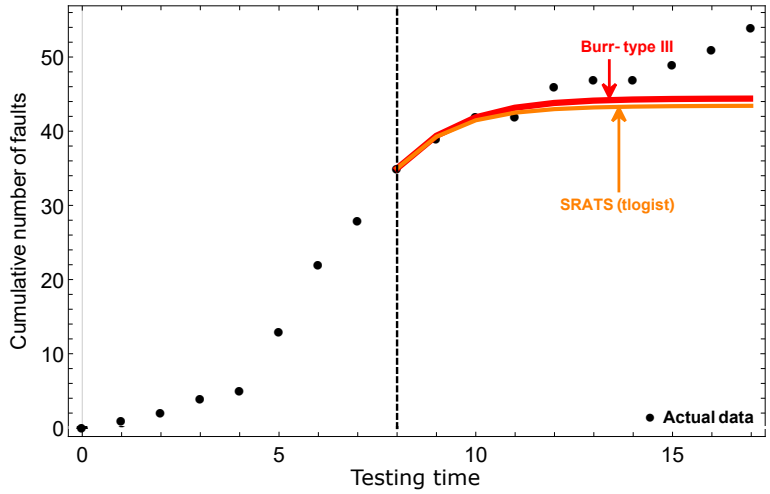


(c) 80% observation point.

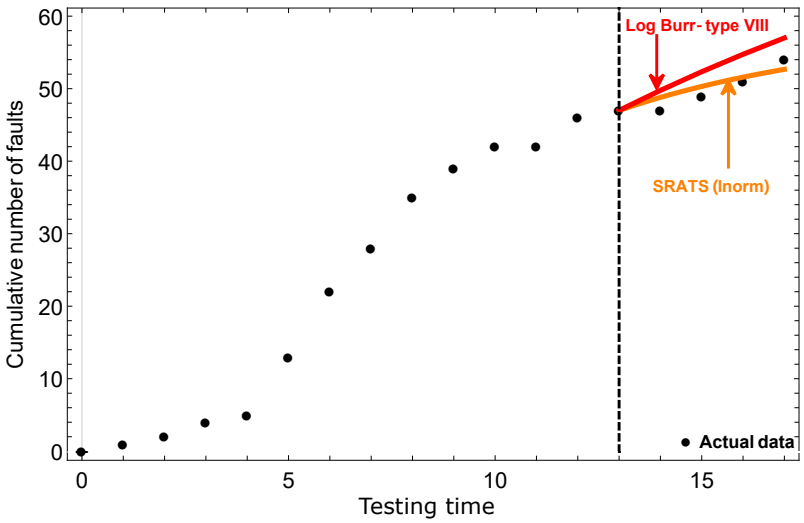
Figure 3.13: Predictive behavior of the cumulative number of software faults with the Burr-type and existing SRMs in TDDS1.



(a) 20% observation point.



(b) 50% observation point.



(c) 80% observation point.

Figure 3.14: Predictive behavior of the cumulative number of software faults with the Burr-type and existing SRMs in TIDS1.

Table 3.26: Predictive performances based on PMSE (time-domain data).

(i) Prediction from the 20% observation point						
Data	**Type-I Burr-type**		**Type-II Burr-type**		**Existing NHPP**	
Set	Best SRM	PMSE	Best SRM	PMSE	Best SRM	PMSE
TDDS1	Burr-type III	**3.951**	Burr-type XII	19.991	Lxvmax	5.073
TDDS2	Burr-type XII	270.380	Log Burr-type IX	141.815	Tnorm	**42.104**
TDDS3	Log Burr-type VI	**28.422**	Burr-type XII	650.939	Lxvmax	32.131
TDDS4	Burr-type III	59.027	Burr-type XI	91.960	Lnorm	**56.477**
TDDS5	Tru Burr-type VI	2022.990	Log Burr-type VIII	**789.554**	Exp	9177.670
TDDS6	Burr-type XII	51.861	Burr-type XII	**50.112**	Txvmin	83.964
TDDS7	Log Burr-type VII	88.338	Log Burr-type VII	193.368	Lxvmax	**32.217**
TDDS8	Burr-type III	2401.850	Burr-type III	**1700.110**	Lxvmax	1852.520
(ii) Prediction from the 50% observation point						
TDDS1	Log Burr-type VI	9.009	Burr-type XII	**4.501**	Pareto	6.118
TDDS2	Tru Burr-type IX	3.749	Tru Burr-type VIII	**0.717**	Tlogist	14.890
TDDS3	Tru Burr-type IX	259.988	Tru Burr-type VIII	475.366	Pareto	**11.712**
TDDS4	Tru Burr-type VIII	121.671	Tru Burr-type VII	**94.254**	Tlogist	103.504
TDDS5	Log Burr-type VIII	252.978	Tru Burr-type VI	**110.137**	Llogist	193.903
TDDS6	Log Burr-type VIII	**4.358**	Burr-type XII	7.727	Lxvmax	10.493
TDDS7	Tru Burr-type IX	**37.702**	Burr-type XII	3909.47	Exp	4480.620
TDDS8	Tru Burr-type VI	**194.841**	Log Burr-type IX	361.550	Lxvmax	32375.500
(iii) Prediction from the 80% observation point						
TDDS1	Log Burr-type VIII	5.720	Tru Burr-type VI	**5.050**	Lxvmax	5.772
TDDS2	Log Burr-type VIII	**0.583**	Burr-type XII	0.590	Lxvmax	0.588
TDDS3	Tru Burr-type VI	**7.997**	Burr-type XII	15.694	Lxvmax	9.419
TDDS4	Log Burr-type IX	**2.867**	Tru Burr-type VIII	4.742	Txvmin	4.253
TDDS5	Tru Burr-type VI	32.079	Burr-type XII	59.795	Lxvmax	**21.715**
TDDS6	Burr-type III	**2.032**	Burr-type XII	2.753	Lxvmax	2.041
TDDS7	Burr-type XII	9.161	Log Burr-type IX	**5.879**	Lxvmax	10.498
TDDS8	Burr-type XII	46.212	Tru Burr-type VII	**20.561**	Lxvmax	57.901

Table 3.27: Predictive performances based on PMSE (group data).

(i) Prediction from the 20% observation point						
Data	**Type-I Burr-type**		**Type-II Burr-type**		**Existing NHPP**	
Set	Best SRM	PMSE	Best SRM	PMSE	Best SRM	PMSE
TIDS1	Burr-type XII	219.067	Burr-type XII	**205.956**	Gamma	220.732
TIDS2	Log Burr-type VI	9.149	Burr-type XII	**5.784**	Lxvmax	29.244
TIDS3	Log Burr-type IX	429.795	Tru Burr-type IX	**330.014**	Gamma	820.049
TIDS4	Burr-type XII	791.335	Tru Burr-type IX	**129.764**	Exp	142.854
TIDS5	Log Burr-type VII	3.535	Tru Burr-type IX	**0.275**	Pareto	2.628
TIDS6	Log Burr-type IX	41.897	Burr-type XII	**30.254**	Tlogist	98.903
TIDS7	Tru Burr-type IX	552.996	Tru Burr-type IX	**233.833**	Exp	387.694
TIDS8	Tru Burr-type VI	**423.360**	Tru Burr-type IX	964.738	Txvmin	448.935
(ii) Prediction from the 50% observation point						
TIDS1	Burr-type III	26.557	Log Burr-type IX	**18.145**	Tlogist	157.837
TIDS2	Burr-type XII	45.793	Log Burr-type VII	**15.789**	Txvmin	30.786
TIDS3	Log Burr-type VIII	**346.721**	Tru Burr-type VIII	838.510	Lxvmax	564.782
TIDS4	Log Burr-type VIII	340.914	Tru Burr-type VIII	**80.646**	Exp	**101.303**
TIDS5	Log Burr-type IX	0.300	Log Burr-type VI	**0.094**	Exp	0.344
TIDS6	Log Burr-type IX	327.310	Tru Burr-type IX	**28.673**	Pareto	365.493
TIDS7	Burr-type III	**22.561**	Tru Burr-type IX	32.780	Lxvmax	22.894
TIDS8	Tru Burr-type IX	**20.613**	Tru Burr-type VIII	307.124	Txvmin	29.110
(iii) Prediction from the 80% observation point						
TIDS1	Log Burr-type VIII	4.676	Burr-type XII	19.559	Lnorm	**1.762**
TIDS2	Log Burr-type VI	**0.455**	Log Burr-type VII	0.802	Exp	0.464
TIDS3	Tru Burr-type IX	**0.230**	Burr-type XII	0.232	Tnorm	0.331
TIDS4	Tru Burr-type IX	**0.695**	Tru Burr-type IX	5.532	Tnorm	1.850
TIDS5	Log Burr-type VI	0.152	Log Burr-type VII	**0.070**	Tnorm	0.224
TIDS6	Burr-type III	1.710	Log Burr-type IX	**1.315**	Lnorm	3.432
TIDS7	Burr-type X	21.403	Tru Burr-type IX	71.140	Txvmin	**6.118**
TIDS8	Log Burr-type VIII	**0.862**	Log Burr-type IX	9.686	Lxvmax	0.864

out of 8 cases. In the group data analysis (see Table 3.27), at 20% observation point, our type-II Burr-type NHPP-based SRMs outperformed the other SRMs in 7 out of 8 data sets. Note that this advantage diminishes as the testing process progresses (5 cases at 50% observation point, 2 cases at 80% observation point). But similar to the experimental results in Table 3.26, our Burr-type SRMs can also outperform the existing NHPPs in almost all datasets.

It should be noted that the best SRM with the minimum PMSE depends on the data sets in modeling frameworks; the type-I Burr, the type-II Burr and the existing NHPP. Of course, the best SRM with the minimum PMSE cannot be known in advance at each observation point. In this sense, we have to say that the comparison in Tables 3.26 and 3.27 is not feasible at each prediction point. In Tables 3.28∼3.30, we compare the predictive performances of SRMs with the minimum AIC at each observation point in the time-domain data sets. In the time-domain data, only in TDDS5 and TDDS7, our type-II Burr-type SRMs could show both the smaller AIC and PMSE. This case also appeared in TDDS2 of the existing NHPP. When the testing phase is middle (50%), the Burr-type NHPP-based SRMs could provide both the smaller AIC and smaller PMSE than the existing NHPP-based SRMs in TDDS7 and TDDS8, and no existing NHPP-based SRMs could realize the similar results. When the testing phase is later (80%), the Burr-type NHPP-based SRMs could show the best goodness-of-fit performance in the observation phase and ensure the minimum PMSE in the future prediction phase in TDDS1, TDDS7, and TDDS8.

For group data, in Tables 3.31∼3.33, it is observed that the Burr-type NHPP-based SRMs provided both the smaller AIC and smaller PMSE at the same time in some cases; *i.e.*, 4 cases out of 8 data sets in (ii) and 4 cases out of 8 data sets in (iii). These results confirm that the Burr-type NHPP-based SRMs have the higher prediction ability than the existing NHPP-based SRMs, especially in the late software testing phase.

In both time-domain and group data sets, when we compare the PMSEs between the best Burr-type NHPP-based SRM and the best existing NHPP-based SRM, we find out that our Burr-type NHPP-based SRMs could guarantee smaller PMSEs than the existing NHPP-based SRMs in many cases; 10 out of 16 cases at 20% observation, 14 out of 16 cases at 50% observation and 12 out

of 16 cases at 80% observation.

We never claim here that the Burr-type NHPP-based SRMs are always better than the existing SRMs in the literature. However, we emphasize that the Burr-type NHPP-based SRMs should be the possible candidates in selecting the best SRM in terms of goodness-of-fit and predictive performances. Also, another new finding is that the logarithmic and truncated Burr-type NHPP-based SRMs gave better goodness-of-fit and prediction results in many cases than the existing Burr-type III, X, and XII SRMs. This would be useful to assuming the competitors of SRMs.

Table 3.28: Predictive performances based on AIC (time-domain data from 20% observation point).

Data	**Type-I Burr-type**			**Type-II Burr-type**			**Existing NHPP**		
Set	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE
TDDS1	Burr-type X	141.639	5252.410	Tru Burr-type VII	141.620	**23.863**	Exp	**141.609**	34.435
TDDS2	Burr-type XII	91.195	270.380	Burr-type XII	89.395	167.194	Tnorm	**84.722**	**42.104**
TDDS3	Log Burr-type IX	**313.458**	885.923	Burr-type XII	313.745	**650.939**	Llogist	313.745	1059.240
TDDS4	Log Burr-type IX	126.424	**64.510**	Tru Burr-type VIII	125.898	224.473	Exp	**121.858**	387.312
TDDS5	Tru Burr-type IX	113.664	37162.600	Tru Burr-type VII	**113.371**	**9562.370**	Exp	113.372	9711.670
TDDS6	Tru Burr-type VIII	128.996	273.412	Log Burr-type VII	**122.741**	589.969	Lxvmax	128.656	**167.226**
TDDS7	Tru Burr-type IX	189.504	89821.400	Tru Burr-type VII	**187.583**	**778.397**	Exp	**187.583**	794.703
TDDS8	Tru Burr-type IX	440.966	6.355E+05	Tru Burr-type IX	**437.627**	5.111E+07	Exp	440.510	**1.399E+05**

Table 3.29: Predictive performances based on AIC (time-domain data from 50% observation point).

Data	**Type-I Burr-type**			**Type-II Burr-type**			**Existing NHPP**		
Set	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE
TDDS1	Burr-type X	403.494	167.253	Tru Burr-type VIII	403.483	**65.333**	Exp	**403.368**	103.723
TDDS2	Log Burr-type VIII	256.977	219.776	Tru Burr-type VIII	256.273	**0.717**	Exp	**256.074**	14.890
TDDS3	Log Burr-type IX	**861.677**	1092.960	Burr-type XII	861.952	**908.166**	Llogist	861.949	960.575
TDDS4	Log Burr-type IX	**334.737**	580.186	Log Burr-type IX	334.749	723.123	Exp	334.762	**106.263**
TDDS5	Log Burr-type IX	364.639	**310.799**	Tru Burr-type VIII	364.127	374.549	Exp	**363.831**	381.855
TDDS6	Log Burr-type VIII	344.810	**4.358**	Burr-type XII	344.653	7.727	Lxvmax	**344.604**	10.493
TDDS7	Tru Burr-type IX	**445.193**	**37.702**	Tru Burr-type VII	446.960	4285.830	Tlogist	445.247	3.924E+06
TDDS8	Burr-type X	1094.480	1.651E+05	Burr-type X	**1092.710**	**5877.410**	Exp	**1092.710**	2.349E+05

Table 3.30: Predictive performances based on AIC (time-domain data from 80% observation point).

Data	**Type-I Burr-type**			**Type-II Burr-type**			**Existing NHPP**		
Set	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE
TDDS1	Log Burr-type VIII	**691.675**	**5.720**	Burr-type XII	692.075	6.422	Lxvmax	691.677	5.772
TDDS2	Log Burr-type VIII	443.895	**0.583**	Tru Burr-type VII	**441.508**	8.711	Lxvmax	443.891	0.588
TDDS3	Log Burr-type IX	**1478.400**	25.476	Burr-type XII	1478.680	**15.694**	Llogist	1478.500	26.709
TDDS4	Log Burr-type X	566.040	**2.867**	Tru Burr-type VIII	565.700	4.742	Exp	**565.497**	4.253
TDDS5	Log Burr-type IX	**577.096**	62.066	Burr-type XII	577.613	59.795	Llogist	577.358	**51.671**
TDDS6	Log Burr-type VII	553.819	5.711	Burr-type XII	553.706	2.753	Lxvmax	**553.472**	**2.041**
TDDS7	Tru Burr-type IX	769.494	16.783	Tru Burr-type VI	**766.032**	**8.464**	Exp	769.836	20.733
TDDS8	Tru Burr-type IX	1886.600	130.770	Tru Burr-type VIII	**1883.940**	**40.017**	Pareto	1889.040	148.226

Table 3.31: Predictive performances based on AIC (group data from 20% observation point).

Data	**Type-I Burr-type**			**Type-II Burr-type**			**Existing NHPP**		
Set	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE
TIDS1	Tru Burr-type IX	12.704	**225.282**	Tru Burr-type VII	10.862	475.805	Exp	**11.085**	625.983
TIDS2	Log Burr-type IX	12.865	**19.943**	Log Burr-type VII	**10.865**	35.183	Lxvmax	12.865	29.244
TIDS3	Log Burr-type IX	18.976	**429.795**	Tru Burr-type VIII	**16.537**	2788.330	Exp	18.442	1860.480
TIDS4	Tru Burr-type VIII	12.649	798.003	Tru Burr-type VIII	**10.649**	418.047	Exp	11.669	**142.854**
TIDS5	Log Burr-type VII	8.000	3.535	Log Burr-type VIII	**6.000**	3.506	Exp	7.386	**2.628**
TIDS6	Tru Burr-type VIII	20.698	490.855	Log Burr-type IX	22.090	**35.594**	Lnorm	**20.660**	478.505
TIDS7	Log Burr-type III	17.340	1251.860	Log Burr-type VI	20.589	**871.373**	Txvmin	**16.958**	1264.850
TIDS8	Burr-type X	8.614	1799.630	Burr-type X	**6.614**	1574.600	Txvmin	8.614	**423.360**

Table 3.32: Predictive performances based on AIC (group data from 50% observation point).

Data	**Type-I Burr-type**			**Type-II Burr-type**			**Existing NHPP**		
Set	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE
TIDS1	Tru Burr-type IX	**34.496**	**38.234**	Tru Burr-type IX	44.792	311.452	Lxvmin	36.846	209.864
TIDS2	Log Burr-type IX	29.849	48.829	Log Burr-type VIII	**28.511**	**18.134**	Lxvmax	31.051	63.385
TIDS3	Log Burr-type VI	**48.466**	**555.562**	Tru Burr-type VIII	49.444	838.510	Exp	49.313	1050.130
TIDS4	Tru Burr-type IX	**30.493**	415.168	Tru Burr-type IX	32.641	**270.063**	Tlogist	30.560	4310.030
TIDS5	Log Burr-type IX	17.787	**0.300**	Log Burr-type VIII	**16.100**	1.153	Exp	17.365	0.344
TIDS6	Log Burr-type IX	**38.584**	327.310	Log Burr-type IX	40.674	**215.080**	Lxvmax	40.521	384.078
TIDS7	Log Burr-type IX	**71.671**	**22.810**	Burr-type XII	72.522	54.804	Lxvmax	72.390	22.894
TIDS8	Tru Burr-type VI	**63.463**	46.071	Burr-type XII	70.374	17840.800	Txvmin	65.835	**29.110**

Table 3.33: Predictive performances based on AIC (group data from 80% observation point).

Data	**Type-I Burr-type**			**Type-II Burr-type**			**Existing NHPP**		
Set	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE	Best SRM	AIC	PMSE
TIDS1	Tru Burr-type IX	**55.028**	**12.006**	Burr-type XII	62.071	19.559	Lxvmin	56.861	12.076
TIDS2	Burr-type IX	50.295	0.836	Log Burr-type VII	**47.200**	**0.802**	Lxvmax	52.523	0.927
TIDS3	Log Burr-type VI	**73.981**	3.123	Tru Burr-type VI	76.515	**2.211**	Txvmin	75.292	4.917
TIDS4	Log Burr-type VI	43.129	**1.371**	Tru Burr-type VIII	46.902	20431.200	Txvmin	**42.540**	3.656
TIDS5	Log Burr-type IX	24.517	1.998	Log Burr-type VII	**22.769**	0.992	Exp	24.271	**0.436**
TIDS6	Burr-type XII	93.323	1.710	Log Burr-type IX	**91.223**	**1.315**	Lxvmax	96.179	5.570
TIDS7	Log Burr-type IX	**112.543**	118.689	Burr-type XII	113.259	98.572	Txvmin	112.836	**6.118**
TIDS8	Tru Burr-type IX	**100.305**	**5.942**	Burr-type XII	126.138	18.125	Tlogist	100.325	5.970

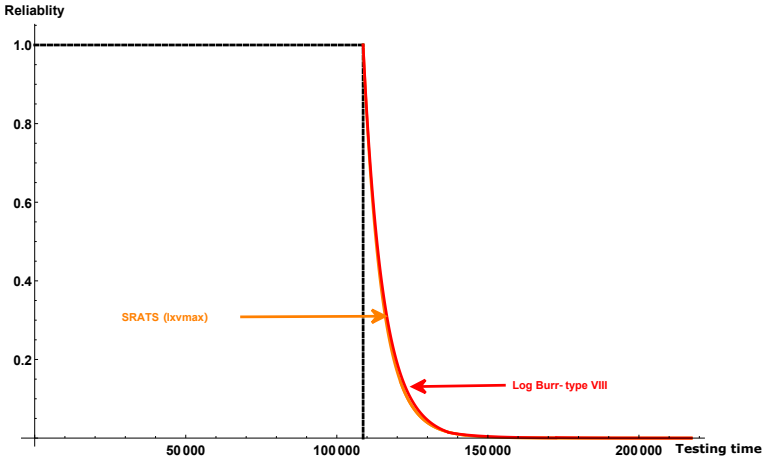
3.3.4.3 Software Reliability Assessment

Finally, we evaluate the software reliability quantitatively with our Burr-type NHPP-based SRMs and compare them with the existing NHPP-based SRMs in SRATS. Let $R(x)$ be the software reliability with the software operational period (prediction length) $x = t_{m+l} - t_m$ or l when the software is released at time $t = t_m$. Since $R(x)$ is defined as the probability that software is fault-free during the time interval $(t, t + x]$, it is easily obtain that

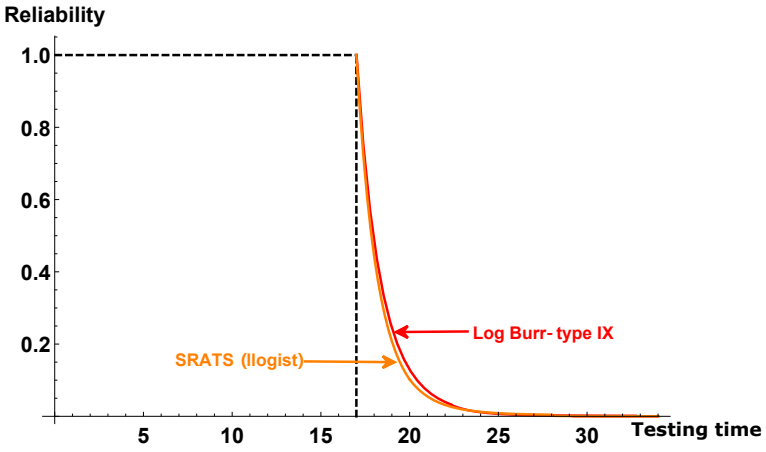
$$\begin{aligned} R(x) &= \Pr(N(t+x) - N(t) = 0 \mid N(t) = m) \\ &= \exp\left(-\left[M(t+x;\boldsymbol{\theta}) - M(t;\boldsymbol{\theta})\right]\right), \end{aligned} \tag{3.36}$$

where m is the cumulative number of software faults detected up to time t in the time-domain data (m in Equation 3.36 is replaced by n_m in the group data). In our subsequent examples, we suppose that the prediction length x is equivalent to the testing length experienced before, say, $t = x$.

Tables 3.34 and 3.35 present the quantitative software reliability. We assume the Burr-type NHPP-based SRM and the SRATS NHPP-based SRM with the minimum AIC in the fault-detection time-domain and group data sets, where the bold font denotes the case with a greater reliability estimate. Looking at these results, it is seen that our Burr-type NHPP-based SRMs could show larger software reliability estimates than the existing NHPP-based SRMs in 3 out of 8 cases (time-domain data) and 4 out of 8 cases (group data). This feature tells us that the Burr-type NHPP-based SRMs tend to make more optimistic decisions in software reliability assessment than the SRATS NHPP-based SRMs. It is worth noting in all the data sets that after each observation point, software faults were additionally detected as the ex-post results. Hence, the optimistic reliability estimation is not preferable. Figure 3.15 (a) and (b) show the software reliability estimates with the Burr-type NHPP-based SRM and the SRATS NHPP-based SRM in TDDS1 and TIDS1, respectively. In both cases, the software reliability values dropped down to zero level rapidly, and two NHPP-based SRMs showed similar reliability values as well. From these results, we find out that both SRMs gave the false alarm to release the current software at respective observation points and request more testing for attaining the requirement level of software reliability.



(a) TDDS1.



(b) TIDS1.

Figure 3.15: Predictive software reliability assessment with the best Burr-type and SRATS NHPP-based SRMs.

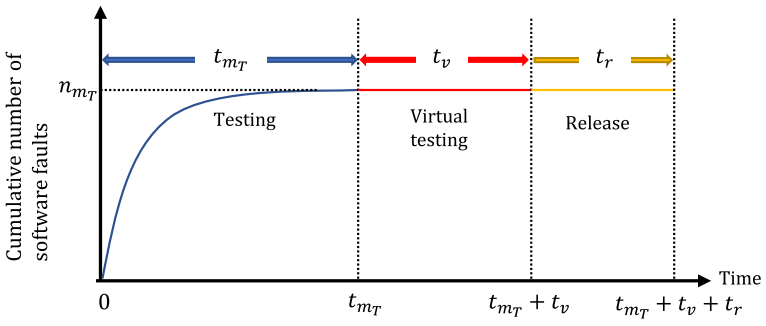


Figure 3.16: Software release decision based on virtual testing time.

Table 3.34: Software reliability assessment with the best AIC (time-domain data).

	Burr-type		**Existing NHPP**	
	Best Burr	**Reliability**	**Best SRM**	**Reliability**
TDDS1	Log Burr-type VIII	2.631E-06	Lxvmax	**2.674E-06**
TDDS2	Log Burr-type VIII	3.687E-03	Lxvmax	**3.751E-03**
TDDS3	Burr-type X	**4.592E-05**	Lxvmin	2.516E-10
TDDS4	Tru Burr-type IX	1.244E-01	Pareto	**1.000E-00**
TDDS5	Burr-type X	**1.035E-05**	Exp	2.596E-08
TDDS6	Log Burr-type VIII	3.283E-04	Lxvmax	**4.694E-03**
TDDS7	Log Burr-type VII	**2.453E-04**	Lxvmax	2.398E-04
TDDS8	Tru Burr-type VI	6.158E-10	Pareto	**7.736E-06**

Next, we introduce the concept of *virtual testing time*, whose idea comes from Zhao et al. [82] to consider a more realistic and plausible software release decision. When the software testing is terminated at a given observed time point, we set the so-called virtual testing time period when no software fault is found. If this hypothesis is correct, we check whether the software reliability can achieve a given requirement level at the end of the virtual testing time and release the software product with a satisfactory level at the end of the virtual testing time period. Otherwise, *i.e.,* if at least one software fault was found, we reset the observation point to the fault detection/fixing time and redefine the virtual testing time from that point. Under the hypothesis that no fault is found during the virtual testing time, the maximum likelihood estimation is made with zero fault count. In the time-domain data, the likelihood function is given by

$$\mathcal{L}(\boldsymbol{\theta};\boldsymbol{D}) = \exp(-M(t_{m_T}+t_v;\boldsymbol{\theta}))\prod_{i=1}^{m_T} M(t_i;\boldsymbol{\theta}), \tag{3.37}$$

where t_v is the virtual testing time (see Figure 3.16). In the group data, the likelihood function is given by

$$\mathcal{L}(\boldsymbol{\theta};\boldsymbol{I}) = \prod_{i=1}^{m_G+v}\left[\frac{[M(t_i;\boldsymbol{\theta})-M(t_{i-1};\boldsymbol{\theta})]^{n_i-n_{i-1}}}{(n_i-n_{i-1})!}\right] \times e^{-[M(t_i;\boldsymbol{\theta})-M(t_{i-1};\boldsymbol{\theta})]}, \tag{3.38}$$

Table 3.35: Software reliability assessment with the best AIC (group data).

	Burr-type		**Existing NHPP**	
	Best Burr	**Reliability**	**Best SRM**	**Reliability**
TIDS1	Log Burr-type IX	**1.065E-02**	Llogist	4.152E-03
TIDS2	Log Burr-type IX	1.353E-05	Lxvmax	**7.236E-05**
TIDS3	Log Burr-type VI	3.751E-02	Tnorm	**3.865E-02**
TIDS4	Tru Burr-type IX	**4.504E-01**	Tlogist	2.816E-01
TIDS5	Log Burr-type IX	**6.548E-03**	Exp	9.832E-04
TIDS6	Log Burr-type IX	1.928E-08	Lxvmax	**1.939E-07**
TIDS7	Burr-type III	8.667E-01	Txvmin	**9.633E-01**
TIDS8	Log Burr-type IX	**6.679E-01**	Llogist	6.373E-01

where $(t_i, n_i) = (t_{m_i}, n_{m_G})$ $(i = m_I + 1, m_I + 2, \ldots, m_I + v)$ with the virtual testing time length v (integer value). Hence, it is obvious that the maximum likelihood estimates depend on the length of virtual testing time t_v or v so increasing virtual testing time leads to increasing quantitative software reliability. Then the problem is to determine an appropriate virtual testing time (t_v^* or v^*) satisfying that the software reliability with a given operational period is greater than a specified requirement lever, *e.g.,* such as 90%.

In our numerical experiments, we focus on the time-domain and group data and set 15 different lengths of virtual testing time (10% to 150% of t_{m_D} (t_{m_I})), where the operational period x is given by $x = t_r = t_{m_D}$ or t_{m_I} and each t_{m_D} or t_{m_I} is given in Table 1.1 (i) and (ii). Tables 3.36, 3.37, 3.37 and 3.38 present the software reliability prediction with the Burr-type NHPP-based SRM and SRATS NHPP-based SRM with the minimum AIC in the time-domain data and group data, when the virtual testing time is given by 10% to 150% length of the testing time t_{m_T} or t_{m_G}. In almost all cases, it is seen that the longer the virtual testing period, the closer the software reliability value to unity. Based on the assumption that no software fault is found during the virtual testing time, NHPP-based SRMs could provide much higher software reliability estimates than the results in Tables 3.34 and 3.35. In other words, it is impossible to guarantee a satisfactory software reliability estimate without setting up the virtual testing time long enough, which implies the belief that the software

product is reliable. In Tables 3.36, 3.37, 3.37 and 3.38, we seek the virtual testing time when the specified reliability level is given by 90%. For instance, in TIDS1, we find that the virtual testing time when the reliability is greater than 90% becomes 130% and 180% of the testing time for the Burr-type NHPP-based SRM and the SRATS NHPP-base SRM, respectively, so that longer virtual testing time with zero fault count than the testing length is required to achieve the requirement because the quantitative software reliability itself is the belief by the tester. In some cases, it is seen that 90% software reliability requirement seems to be unrealistic because the virtual testing time with zero fault count must be dozens of times t_{m_D} or t_{m_I}, *i.e.*, in TDDS1, TDDS2, TDDS6, TDDS7, and TIDS6. On one hand, in TDDS4 and TIDS7, the 90% requirement level is achieved after the software testing when SRATS NHPP-based SRM is used. Of course, the software release decision considered here is based on the existence of virtual testing time with zero fault count. If any software fault was detected during the period, the observation point to trigger the virtual testing is changed step by step.

Table 3.36: Reliability prediction with virtual testing time in time-domain data (TDDS1 $\sim$ TDDS4).

	TDDS1		**TDDS2**		**TDDS3**		**TDDS4**	
Virtual testing time	**Burr-type (Log Burr-type VIII)**	**Existing NHPP (Lxvmax)**	**Burr-type (Log Burr-type VIII)**	**Existing NHPP (Lxvmax)**	**Burr-type (Burr-type X)**	**Existing NHPP (Lxvmin)**	**Burr-type Tru Burr-type IX**	**Existing NHPP (Pareto)**
10%	2.648E-05	2.778E-05	8.831E-03	9.117E-03	6.174E-02	1.110E-06	4.284E-01	1.000E+00
20%	1.434E-04	1.518E-04	1.728E-02	1.787E-02	3.267E-01	1.512E-04	9.717E-01	1.000E+00
30%	5.167E-04	5.465E-04	2.924E-02	3.022E-02	6.260E-01	2.981E-03	1.000E+00	1.000E+00
40%	1.407E-03	1.480E-03	4.458E-02	4.597E-02	8.209E-01	1.933E-02	1.000E+00	1.000E+00
50%	3.189E-03	1.847E-05	6.293E-02	6.472E-02	9.213E-01	6.467E-02	1.000E+00	1.000E+00
60%	6.049E-03	6.244E-03	8.371E-02	8.591E-02	9.675E-01	8.591E-02	1.000E+00	1.000E+00
70%	1.050E-02	1.064E-02	1.068E-01	1.090E-01	9.872E-01	2.486E-01	1.000E+00	1.000E+00
80%	1.616E-02	1.664E-02	1.305E-01	1.334E-01	9.953E-01	3.626E-01	1.000E+00	1.000E+00
90%	2.371E-02	2.431E-02	1.555E-01	1.587E-01	9.983E-01	4.733E-01	1.000E+00	1.000E+00
100%	3.291E-02	3.365E-02	1.811E-01	1.844E-01	9.994E-01	5.726E-01	1.000E+00	1.000E+00
110%	4.370E-02	4.455E-02	2.067E-01	2.102E-01	9.998E-01	6.575E-01	1.000E+00	1.000E+00
120%	5.594E-02	5.694E-02	2.323E-01	2.359E-01	9.999E-01	7.278E-01	1.000E+00	1.000E+00
130%	6.950E-02	7.060E-02	2.571E-01	2.612E-01	1.000E+00	7.847E-01	1.000E+00	1.000E+00
140%	8.418E-02	8.536E-02	2.822E-01	2.859E-01	1.000E+00	8.302E-01	1.000E+00	1.000E+00
150%	9.981E-02	1.011E-01	3.064E-01	3.101E-01	1.000E+00	8.663E-01	1.000E+00	1.000E+00
Virtual test time required to reach 90% reliability	1550%	1550%	1110%	1100%	50%	270%	20%	0%

Table 3.37: Reliability prediction with virtual testing time in time-domain data (TDDS5 $\sim$ TDDS8).

	TDDS5		**TDDS6**		**TDDS7**		**TDDS8**	
Virtual testing time	**Burr-type (Burr-type X)**	**Existing NHPP (Exp)**	**Burr-type (Log Burr-type VIII)**	**Existing NHPP (Lxvmax)**	**Burr-type (Log Burr-type VII)**	**Existing NHPP (Lxvmax)**	**Burr-type (Tru Burr-type VI)**	**Existing NHPP (Pareto)**
10%	2.759E-02	2.892E-05	1.597E-03	1.566E-02	1.157E-03	1.144E-03	3.153E-04	6.944E-05
20%	2.505E-01	1.285E-03	2.433E-03	3.586E-02	3.657E-03	3.534E-03	8.535E-04	3.586E-04
30%	5.606E-01	1.197E-02	5.290E-03	6.520E-02	8.312E-03	8.267E-03	4.304E-03	1.256E-03
40%	7.797E-01	4.812E-02	1.494E-02	1.020E-01	1.595E-02	1.591E-02	7.400E-01	3.381E-03
50%	8.984E-01	1.190E-01	2.660E-02	1.439E-01	2.686E-02	2.683E-02	9.765E-01	4.069E-03
60%	9.557E-01	2.187E-01	3.086E-02	1.888E-01	4.097E-02	4.096E-02	9.975E-01	1.404E-02
70%	9.815E-01	3.326E-01	5.698E-02	2.348E-01	5.802E-02	5.801E-02	9.999E-01	2.369E-02
80%	9.927E-01	4.466E-01	6.776E-02	2.803E-01	7.760E-02	7.757E-02	1.000E+00	3.659E-02
90%	9.972E-01	5.513E-01	7.570E-02	3.246E-01	9.921E-02	9.925E-02	1.000E+00	5.270E-02
100%	9.990E-01	6.420E-01	1.299E-01	3.671E-01	1.224E-01	1.224E-01	1.000E+00	7.189E-02
110%	9.997E-01	7.176E-01	1.508E-01	4.072E-01	1.465E-01	1.467E-01	1.000E+00	9.349E-02
120%	9.999E-01	7.790E-01	1.710E-01	4.450E-01	1.664E-01	1.718E-01	1.000E+00	1.173E-01
130%	1.000E+00	8.281E-01	2.021E-01	4.802E-01	1.896E-01	1.971E-01	1.000E+00	1.429E-01
140%	1.000E+00	8.667E-01	2.312E-01	5.130E-01	2.157E-01	2.226E-01	1.000E+00	1.696E-01
150%	1.000E+00	8.970E-01	2.331E-01	5.435E-01	2.328E-01	2.478E-01	1.000E+00	1.972E-01
Virtual test time required to reach 90% reliability	60%	160%	1240%	560%	1390%	1070%	50%	880%

Table 3.38: Reliability prediction with virtual testing time in group data (TIDS1 ∼ TIDS4).

	TIDS1		**TIDS2**		**TIDS3**		**TIDS4**	
Virtual testing time	**Burr-type (Log Burr-type IX)**	**Existing NHPP (Llogist)**	**Burr-type (Log Burr-type IX)**	**Existing NHPP (Lxvmax)**	**Burr-type (Log Burr-type VI)**	**Existing NHPP (Tnorm)**	**Burr-type (Tru Burr-type IX)**	**Existing NHPP (Tlogist)**
10%	3.846E-02	1.749E-02	5.844E-05	3.721E-04	1.433E-01	1.918E-01	7.148E-01	5.470E-01
20%	1.518E-01	8.713E-02	1.588E-04	1.232E-03	5.084E-01	6.276E-01	8.621E-01	7.384E-01
30%	3.073E-01	2.003E-01	7.201E-04	6.245E-03	7.662E-01	8.738E-01	**9.159E-01**	8.543E-01
40%	3.785E-01	2.630E-01	1.232E-03	1.106E-02	**9.139E-01**	**9.629E-01**	9.685E-01	**9.200E-01**
50%	5.137E-01	3.849E-01	6.354E-03	2.600E-02	9.596E-01	9.901E-01	9.919E-01	9.761E-01
60%	6.215E-01	4.926E-01	1.315E-02	3.605E-02	9.830E-01	9.976E-01	9.970E-01	9.869E-01
70%	6.669E-01	5.331E-01	2.322E-02	4.763E-02	9.939E-01	9.995E-01	9.986E-01	9.928E-01
80%	7.055E-01	6.208E-01	5.250E-02	7.499E-02	9.970E-01	9.999E-01	9.994E-01	9.961E-01
90%	7.877E-01	6.864E-01	7.220E-02	8.968E-02	9.991E-01	1.000E+00	9.997E-01	9.978E-01
100%	8.268E-01	7.390E-01	1.166E-01	1.214E-01	9.996E-01	1.000E+00	1.000E+00	9.994E-01
110%	8.496E-01	7.613E-01	1.408E-01	1.379E-01	9.998E-01	1.000E+00	1.000E+00	9.996E-01
120%	8.710E-01	7.994E-01	1.658E-01	1.546E-01	9.999E-01	1.000E+00	1.000E+00	9.998E-01
130%	**9.002E-01**	8.301E-01	2.170E-01	1.884E-01	1.000E+00	1.000E+00	1.000E+00	9.999E-01
140%	9.151E-01	8.433E-01	2.425E-01	2.051E-01	1.000E+00	1.000E+00	1.000E+00	9.999E-01
150%	9.241E-01	8.660E-01	2.926E-01	2.381E-01	1.000E+00	1.000E+00	1.000E+00	1.000E+00
Virtual test time required to reach 90% reliability	130%	180%	350%	1470%	40%	40%	30%	40%

Table 3.39: Reliability prediction with virtual testing time in group data (TIDS5 ∼ TIDS8).

	TIDS5		**TIDS6**		**TIDS7**		**TIDS8**	
Virtual testing time	**Burr-type (Log Burr-type IX)**	**Existing NHPP (Exp)**	**Burr-type (Log Burr-type IX)**	**Existing NHPP (Lxvmax)**	**Burr-type (Burr-type III)**	**Existing NHPP (Txvmin)**	**Burr-type (Log Burr-type IX)**	**Existing NHPP (Logist)**
10%	3.595E-02	8.250E-03	2.321E-07	4.471E-06	**9.638E-01**	**9.993E-01**	7.970E-01	7.870E-01
20%	2.167E-01	2.717E-02	1.597E-06	3.786E-05	9.902E-01	1.000E+00	8.719E-01	8.752E-01
30%	5.625E-01	5.967E-02	6.094E-06	1.779E-04	9.970E-01	1.000E+00	**9.3709E-01**	**9.204E-01**
40%	4.455E-01	7.138E-02	2.256E-05	5.724E-04	9.993E-01	1.000E+00	9.560E-01	9.489E-01
50%	5.956E-01	8.958E-02	4.912E-05	1.419E-03	9.997E-01	1.000E+00	9.601E-01	9.662E-01
60%	6.607E-01	9.725E-02	1.774E-04	2.928E-03	9.999E-01	1.000E+00	9.791E-01	9.770E-01
70%	6.834E-01	1.045E-01	6.631E-04	5.287E-03	1.000E+00	1.000E+00	9.842E-01	9.840E-01
80%	7.766E-01	1.185E-01	1.811E-03	8.631E-03	1.000E+00	1.000E+00	9.910E-01	9.887E-01
90%	8.494E-01	1.255E-01	3.987E-03	1.304E-02	1.000E+00	1.000E+00	9.955E-01	9.918E-01
100%	8.869E-01	1.401E-01	7.506E-03	1.852E-02	1.000E+00	1.000E+00	9.940E-01	9.940E-01
110%	8.893E-01	1.478E-01	1.260E-02	2.505E-02	1.000E+00	1.000E+00	9.977E-01	9.955E-01
120%	8.957E-01	1.558E-01	1.938E-02	3.257E-02	1.000E+00	1.000E+00	9.983E-01	9.966E-01
130%	**9.298E-01**	1.727E-01	2.785E-02	4.101E-02	1.000E+00	1.000E+00	9.988E-01	9.974E-01
140%	9.309E-01	1.817E-01	3.793E-02	5.023E-02	1.000E+00	1.000E+00	9.991E-01	9.980E-01
150%	9.467E-01	2.007E-01	3.795E-02	6.019E-02	1.000E+00	1.000E+00	9.980E-01	9.980E-01
Virtual test time required to reach 90% reliability	130%	750%	1890%	3380%	10%	0%	30%	30%

3.4 Numerical Comparison between All Parametric NHPP-based SRMs

Finally, we summarize the numerical experimental results of the novel NHPP-based SRMs we proposed in this chapter. A comparison among the existing type-I NHPP-based SRMs, the type-II NHPP-based SRMs with representative fault-detection time distribution, the Lindley-type NHPP-based SRMs, and the Burr-type NHPP-based SRMs is performed to confirm which SRM can guarantee the best goodness-of-fit and predictive performances. In Tables 3.40 and 3.41, we compare the AIC of these 4 types of NHPP-based SRMs in the time-domain data and group data, respectively. It is observed that our Burr-type NHPP-based SRMs outperformed the other SRMs in most cases, in both the time-domain data and group data. The existing type-I NHPP-based SRMs could only guarantee the smaller AIC in TDDS6 and TIDS7. Then, we focus on the PMSE in Tables 3.42 and 3.43 to investigate the predictive performance among 4 types of NHPP-based SRMs. At this point, it is obvious that the Burr-type NHPP-based SRM that could show the smaller AIC, no longer guarantees a smaller PMSE. It indicates that the SRM that performs well in terms of goodness-of-fit does not necessarily have a superior software fault prediction capability. However, there is no denying that software testers still tend to choose SRMs with better goodness-of-fit performance to predict the number of software faults in future phases. Hence, we compare the PMSEs of 4 types of NHPP-based SRMs in Tables 3.42 and 3.43 for the time-domain data and group data, respectively. For the time-domain data, at 50 % and 80 % observation points, the novel NHPP-based SRMs we propose in this chapter could completely outperform the existing type-I NHPP-based SRMs, while even at 20% observation points, the existing type-I NHPPs guaranteed smaller PMSEs in TDDS2, TDDS4, and TDDS7. For the group data, at each observation point, our Lindley-type and Burr-type NHPP-based SRMs were able to guarantee the smaller PMSE at least in general cases at every observation point. Hence, we can conclude that our proposed new NHPP-based SRMs could be essentially replaced by the existing NHPP-based SRMs in many cases.

Table 3.40: Goodness-of-fit performances based on AIC (time-domain data).

	Representative existing type-I NHPP-based SRMs		**Our proposed NHPP-based SRMs in Chapter 3**					
	Best SRM		**Best type-II with representative time distributions**		**Best Lindley-type**		**Best Burr-type**	
Data Set	**AIC**	**MSE**	**AIC**	**MSE**	**AIC**	**MSE**	**AIC**	**MSE**
TDDS1	896.666	1.950	**895.305**	2.315	898.040	2.860	896.663	**1.941**
TDDS2	598.131	1.705	596.501	1.809	600.530	1.680	**595.295**	**1.557**
TDDS3	1938.160	6.570	1939.600	8.052	**1938.200**	**6.000**	1938.650	7.138
TDDS4	759.579	**3.747**	759.948	5.509	759.950	4.160	**754.822**	3.975
TDDS5	757.869	19.985	757.031	19.315	758.530	**16.820**	**755.269**	22.792
TDDS6	**721.928**	**1.442**	726.052	2.803	724.450	2.190	722.397	2.095
TDDS7	1008.220	5.970	1007.100	7.039	1009.400	7.580	**1005.730**	**4.212**
TDDS8	2504.170	47.404	2503.370	63.699	2511.640	58.340	**2475.520**	**13.034**

Table 3.41: Goodness-of-fit performances based on AIC (group data).

	Representative existing type-I NHPP-based SRMs		**Our proposed NHPP-based SRMs in Chapter 3**					
	Best SRM		**Best type-II with representative time distributions**		**Best Lindley-type**		**Best Burr-type**	
Data Set	**AIC**	**MSE**	**AIC**	**MSE**	**AIC**	**MSE**	**AIC**	**MSE**
TIDS1	73.053	4.115	85.339	48.269	84.490	5.520	**72.500**	**3.819**
TIDS2	61.694	3.239	60.674	3.557	63.840	3.640	**54.632**	3.283
TIDS3	87.275	6.151	91.919	31.232	87.270	6.610	**85.873**	**3.086**
TIDS4	51.052	1.968	63.556	27.199	51.050	2.320	**50.256**	**1.816**
TIDS5	29.911	0.118	27.953	0.118	31.22	0.130	**29.132**	0.567
TIDS6	108.831	**22.514**	107.211	24.394	104.280	22.900	**102.871**	34.921
TIDS7	**123.256**	**2.122**	138.029	24.847	126.930	3.810	124.767	2.180
TIDS8	117.470	9.408	148.438	45.178	120.630	10.440	**117.234**	**9.042**

Table 3.42: Predictive performances based on PMSE (time-domain data).

Data Set	Existing type-I NHPP	Type-II NHPP	Lindley-type	Burr-type
(i) Prediction from the 20% observation point				
TDDS1	5.073	6.420	4.610	**3.951**
TDDS2	**42.104**	145.648	229.840	141.815
TDDS3	32.131	1417.110	177.420	**28.422**
TDDS4	**56.477**	198.490	137.850	59.027
TDDS5	9177.670	467.320	**194.330**	789.554
TDDS6	83.960	79.614	80.900	**50.112**
TDDS7	**32.217**	207.592	71.490	88.338
TDDS8	1852.520	1474.020	**723.310**	1700.110
(ii) Prediction from the 50% observation point				
TDDS1	6.118	6.420	10.080	**4.501**
TDDS2	14.890	11.747	3.700	**0.717**
TDDS3	11.712	**10.283**	455.740	259.988
TDDS4	103.504	106.282	**68.140**	94.254
TDDS5	193.903	77.498	**41.100**	110.137
TDDS6	10.493	30.944	18.820	**4.358**
TDDS7	4480.620	**18.425**	322.720	37.702
TDDS8	3569.230	**45.344**	146.700	194.841
(iii) Prediction from the 80% observation point				
TDDS1	5.772	**3.432**	4.440	5.050
TDDS2	0.588	0.819	0.600	**0.583**
TDDS3	9.419	19.992	33.830	**7.997**
TDDS4	4.253	4.258	3.900	**2.867**
TDDS5	21.715	51.677	**5.020**	32.079
TDDS6	2.041	3.697	**1.590**	2.032
TDDS7	10.498	**4.291**	13.470	5.879
TDDS8	57.901	**9.268**	59.170	20.561

Table 3.43: Predictive performances based on PMSE (group data).

Data Set	Existing type-I NHPP	Type-II NHPP	Lindley-type	Burr-type
(i) Prediction from the 20% observation point				
TIDS1	220.732	218.763	**93.000**	205.956
TIDS2	29.244	47.377	36.260	**5.784**
TIDS3	820.049	171.702	**123.010**	300.014
TIDS4	142.854	86.083	**66.570**	129.764
TIDS5	2.628	2.625	0.330	**0.275**
TIDS6	98.903	**25.613**	117.590	30.2254
TIDS7	387.694	67.730	**52.320**	233.833
TIDS8	448.935	423.360	**311.520**	423.360
(ii) Prediction from the 50% observation point				
TIDS1	157.837	159.545	20.740	**18.145**
TIDS2	30.786	**3.722**	14.800	15.789
TIDS3	564.782	849.736	**184.010**	346.721
TIDS4	101.303	101.258	95.310	**80.646**
TIDS5	0.344	0.347	0.140	**0.094**
TIDS6	365.493	**18.825**	20.650	28.673
TIDS7	22.894	27.045	**17.290**	22.561
TIDS8	29.110	156.329	125.440	**20.613**
(iii) Prediction from the 80% observation point				
TIDS1	**1.762**	8.736	3.200	4.676
TIDS2	0.464	0.464	**0.360**	0.455
TIDS3	0.331	41.228	0.340	**0.230**
TIDS4	1.850	18.985	**0.410**	0.695
TIDS5	0.224	0.090	0.080	**0.070**
TIDS6	3.432	6.144	**1.100**	1.315
TIDS7	**6.118**	17.300	30.110	21.403
TIDS8	0.864	6.333	2.690	**0.862**

Chapter 4

NHPP-based Software Reliability Modeling with Local Polynomial Debug Rate

In this chapter, we propose local polynomial SRMs, which can be categorized into a semi-parametric modeling framework. Our models belong to the common NHPP-based SRMs but possess a flexible structure to approximate an arbitrary mean value function by controlling the polynomial degree. More specifically, we develop two types of local polynomial NHPP-based SRMs; type-I and type-II SRMs, which are substantial extensions of the existing NHPP-based SRMs in a similar category.

4.1 Preliminary

As we know, the practical experiences suggest no unique SRM exists, which could fit every software fault-count data, so in parametric software reliability modeling based on the representative software fault-detection time distribution, the selection of the fault-detection time c.d.f. is always required. Since the goodness-of-fit and predictive performances for the parametric SRMs strongly depend on the software fault-count data, it is quite important to apply the so-called semi-parametric SRMs without specifying the software fault-detection time c.d.f. The gamma-mixture NHPP-based SRM [83] can be regarded as an intuitively convinced semi-parametric SRM to unify the existing NHPP-

based SRMs approximately. Okamura and Dohi [84] extended the gamma-mixture NHPP-based SRM [83], and developed the phase-type NHPP-based SRMs by assuming the phase-type distribution in the software fault-detection time c.d.f.. Since the phase-type distribution can approximate an arbitrary c.d.f. with arbitrary accuracy by designing the phase structure of the underlying continuous-time Markov chain, it seems to involve all NHPP-based SRMs in the modeling framework. However, it should be noted that finding the optimal phase structure is almost impossible. So, we still encounter serious problems in determining the phase structure and the model freedom indicating the number of phases when the phase-type NHPP-based SRMs are considered. Nafreen and Fiondella [85] concerned the software debug rate and overviewed several NHPP-based SRMs with bathtub-shaped debug rate by dealing with a low-order local polynomial function called the quadratic model.

In this chapter, the fundamental idea comes from the assumption that the software debug rate, which is equivalent to the hazard rate function of software fault-detection time, is approximated by an arbitrary local polynomial function. This idea seems well-motivated to provide a feasible semi-parametric NHPP-based SRM, because one does need neither to select a parametric c.d.f. nor to determine the phase structure. The main feature of the local polynomial NHPP-based SRMs is to improve the goodness-of-fit by controlling the polynomial degree. More precisely, we determine the polynomial degree by AIC and select the best local polynomial debug rate. We treat the type-I and type-II NHPP-based SRMs as well with high-order local polynomial debug rate and investigate both the goodness-of-fit and predictive performances of our semi-parametric NHPP-based SRMs through comprehensive experiments with actual software development project data.

4.2 Software Debug Rate

4.2.1 Introduction

In addition to viewing the NHPP-based SRMs with the mean value function (see Chapter 3), Yamada and Osaki [86] also characterized the NHPP-based SRMs with the *software debug rate*. It implies the instantaneous fault-detection

rate per fault at time t as follows.

$$d(t;\boldsymbol{\alpha}) = \frac{dM(t;\boldsymbol{\theta})/dt}{\omega - M(t;\boldsymbol{\theta})} = \frac{f(t;\boldsymbol{\alpha})}{\{1 - F(t;\boldsymbol{\alpha})\}}, \tag{4.1}$$

where $f(t;\boldsymbol{\alpha}) = dF(t;\boldsymbol{\alpha})/dt$ is the probability density function (p.d.f.). Hence, the software debug rate is equivalent to the *hazard rate* of the fault-detection time c.d.f. $F(t;\boldsymbol{\alpha})$, as well as the intensity function in Equation (1.12). In Table 1.1, we summarize the c.d.f., mean value function and software debug rate function for the 11 existing type-I NHPP-based SRMs in Table 3.1, and the 3 existing type-II NHPP-based SRMs (Musa-Okumoto, Cox-Lewis and Duane SRMs).

4.2.2 Polynomial Software Debug Rate

Probability distributions with a local polynomial hazard rate function have been used for modeling lifetimes in reliability engineering. Apart from the NHPP-based software reliability modeling, several authors concerned the polynomial hazard rate models in the traditional lifetime data analysis. Bain [87], Balakrishnan and Malik [88], Mahmoud and Al-Nagar [89] considered a low-order polynomial model called the *linear exponential distribution* in the lifetime data analysis. Lawless [90] gave some examples of the least-squares estimation and the maximum likelihood estimation for the fundamental polynomial hazard rate models and their variants with censoring and grouped data. Krane [91] applied the polynomial model to the multiple regression analysis. Kogan [92] proposed a computation algorithm to obtain the moments from the order statistics of lifetime data with generalized bathtub hazard rate. Csenki [93] derived the Laplace transform of the continuous random variable with a local polynomial hazard rate function and applied it to estimate the polynomial coefficients from the sample moments of the c.d.f. Bagkavos and Patil [94] proposed a local polynomial fitting by means of the kernel method in failure rate estimation.

Suppose in the NHPP-based software reliability modeling that

$$d(t;\boldsymbol{\alpha}) = \mu_0 + \mu_1 t + \mu_2 t^2 + \cdots + \mu_m t^m, \tag{4.2}$$

where $\boldsymbol{\alpha} = (\mu_0, \mu_1, \ldots, \mu_m) \in \mathbb{R}^{m+1}$. Then the c.d.f. is expressed as

$$F(t;\boldsymbol{\alpha}) = 1 - \exp\left(-\sum_{j=0}^{m} \frac{\mu_j t^{j+1}}{j+1}\right). \tag{4.3}$$

Table 4.1: The representative existing NHPP-based SRMs.

Type-I NHPP-based SRMs	
Time distribution & SRM	**Debug rate function**
Exponential dist. (Exp)	$d(t;\boldsymbol{\alpha}) = b$
Gamma dist. (Gamma)	$d(t;\boldsymbol{\alpha}) = \frac{c^b t^{b-1} e^{-ct}}{\Gamma(b) - \int_0^t s^{b-1} e^{-cs} ds}$
Pareto dist. (Pareto)	$d(t;\boldsymbol{\alpha}) = \frac{b}{c+t}$
Truncated normal dist. (Tnorm)	$d(t;\boldsymbol{\alpha}) = -\frac{\sqrt{\frac{2}{\pi}} e^{-\frac{(c-t)^2}{2b^2}}}{b\left(\mathrm{erfc}\left(\frac{c-t}{\sqrt{2}b}\right)-2\right)}$
Log-normal dist. (Lnorm)	$d(t;\boldsymbol{\alpha}) = \frac{\sqrt{\frac{2}{\pi}} e^{-\frac{(c-\log(t))^2}{2b^2}}}{bt\left(\mathrm{erfc}\left(\frac{c-\log(t)}{\sqrt{2}b}\right)-2\right)}$
Truncated logistic dist. (Tlogist)	$d(t;\boldsymbol{\alpha}) = \frac{e^{t/b}}{be^{c/b}+be^{t/b}}$
Log-logistic dist. (Llogist)	$d(t;\boldsymbol{\alpha}) = \frac{t^{\frac{1}{b}-1}}{b\left(e^{c/b}+t^{1/b}\right)}$
Truncated extreme-value max dist. (Txvmax)	$d(t;\boldsymbol{\alpha}) = \frac{e^{\frac{c-t}{b}}}{b\left(e^{e^{\frac{c-t}{b}}}-1\right)}$
Log-extreme-value max dist. (Lxvmax)	$d(t;\boldsymbol{\alpha}) = \frac{e^{c/b} t^{-\frac{b+1}{b}}}{b\left(e^{e^{c/b} t^{-1/b}}-1\right)}$
Truncated extreme-value min dist. (Txvmin)	$d(t;\boldsymbol{\alpha}) = \frac{e^{\frac{c+t}{b}}}{b}$
Log-extreme-value min dist. (Lxvmin)	$d(t;\boldsymbol{\alpha}) = \frac{e^{c/b} t^{\frac{1}{b}-1}}{b}$
Type-II NHPP-based SRMs	
Pareto dist. (Muse-Okumoto)	$d(t;\boldsymbol{\alpha}) = \frac{b}{c+t}$
Truncated extreme-value min dist. (Cox-Lewis)	$d(t;\boldsymbol{\alpha}) = e^{b+ct}$
Log-extreme-value min dist. (Duane)	$d(t;\boldsymbol{\alpha}) = \frac{e^{c/b} t^{\frac{1}{b}-1}}{b}$

$\Gamma(\cdot)$: standard gamma function

$\mathrm{erfc}(\cdot)$: complementary error function

$\ln(\cdot)$: natural logarithmic function

The above c.d.f. with $m+1$ degrees is interpreted as a probability model on the minimum of $m+1$ independent Weibull random variables if $\boldsymbol{\alpha} \in \mathbb{R}_+^{m+1}$, where j-th of them has a scale parameter $\sqrt[j+1]{(j+1)/\mu_j}$ and shape parameter $j+1$. The above probability distribution is called the *poly-Weibull distribution*. Berger and Sun [95], Davison and Louzaada-Neto [96] established the Bayesian estimation for the poly-Weibull distribution. Freels et al. [97] considered the maximum likelihood estimation for the poly-Weibull distribution. Demiris et al. [98] applied the poly-Weibull distribution to investigate the effectiveness of cardio-thoracic transplantation in the survivor analysis. However, the related

works mentioned above made the strong assumption of $\boldsymbol{\alpha} \in \mathbb{R}_+^{m+1}$, so that $\boldsymbol{\alpha}$ is a non-negative vector.

In our NHPP-based SRM, once the local polynomial hazard rate function is determined, from Equations (3.1) and (3.5), the mean value functions of the type-I local polynomial NHPP-based SRM and the type-II local polynomial NHPP-based SRM can be obtained as

$$M(t;\boldsymbol{\theta}) = \omega\left(1 - \exp\left(-\sum_{j=0}^{m}\frac{\mu_j t^{j+1}}{j+1}\right)\right) \tag{4.4}$$

and

$$M(t;\boldsymbol{\alpha}) = \sum_{j=0}^{m}\frac{\mu_j t^{j+1}}{j+1}, \tag{4.5}$$

respectively. As the special cases of type-I local polynomial NHPP-based SRM, when $m = 0$ and $m = 1$, the hazard rate functions become $d(t;\boldsymbol{\alpha}) = \mu_0$ and $d(t;\boldsymbol{\alpha}) = \mu_0 + \mu_1 t$, respectively. When $m = 1$, the associated c.d.f. becomes the linear exponential distribution [87, 88, 89]. Nafreen and Fiondella [85] considered a type-I NHPP-based SRM with the linear exponential distribution for the purpose to develop a bathtub-shaped software debug rate. Hence, it is evident that Equation (4.2) is a general form to express the software debug rate.

If we assume that $\boldsymbol{\alpha} \in \mathbb{R}_+^{m+1}$, i.e., $(\mu_0, \mu_1, \ldots, \mu_m)$ are all positive real numbers, it always holds that $d(t;\boldsymbol{\alpha}) \geq 0$ and $F(t;\boldsymbol{\alpha})$ is increasing hazard rate (IHR). However, dissimilar to hardware reliability, it is well known that the software reliability growth phenomenon can be observed in software testing. In other words, the IHR assumption seems to be rather strong and not to be plausible to explain the software reliability growth. Hence the polynomial parameters may be negative except for μ_0, because $\mu_0 \geq 0$ is a necessary condition of $d(t;\boldsymbol{\alpha}) \geq 0$. In fact, it is not so easy to find out $\boldsymbol{\alpha} \in \mathbb{R}^m$ with $\mu_0 \geq 0$ to satisfy the constraint $d(t;\boldsymbol{\alpha}) \geq 0$.

4.3 Parameter Estimation

In this chapter, we consider only the software fault count group data, which consists of the number of detected faults in a set of calendar-time-based intervals $[t_{i-1}, t_i)$ $(i = 1, 2, \ldots, k)$. The likelihood function and log likelihood function are shown in Equations (3.8) and (3.9).

Then the maximum likelihood (ML) estimate in the estimation phase, $\hat{\boldsymbol{\theta}} \in (\hat{\omega}, \hat{\boldsymbol{\alpha}})$, can be obtained as $\text{argmax}_{\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}} \ln \mathcal{L}(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})$ in Equation (3.8), subject to $d(t_i, \hat{\boldsymbol{\alpha}}) = \hat{\mu}_0 + \hat{\mu}_1 t_i + \hat{\mu}_2 t_i^2 + \cdots + \hat{\mu}_m t_i^m \geq 0$ with $i = 1, 2, \ldots, k$.

In what follows, we consider the following two cases;

- **Case I:** $\boldsymbol{\alpha} = (\mu_0, \mu_1, \ldots, \mu_m) \in \mathbb{R}_+^{m+1}$ are all positive real numbers.
- **Case II:** $\mu_0 \geq 0$ and $(\mu_1, \ldots, \mu_m) \in \mathbb{R}^m$ are real numbers.

It is evident that Case I is rather restrictive because the software debug rate is always increasing over time t. However, the maximum likelihood estimation is easily made because of $d(t_i, \hat{\boldsymbol{\alpha}}) \geq 0$ for all the observation data t_i $(i = 1, 2, \ldots, k)$. In Case II, we consider all the combinations of $\mu_j \in \mathbb{R}_+$ and $\mu_j \in \mathbb{R}_-$ for all $j = 1, 2, \ldots, m$, say 2^m combinations, and solve the maximization problems with constraint $d(t_i, \boldsymbol{\alpha}) \geq 0$. Note that the general-purpose optimization solver such as *Mathematica* and *MATLAB* enables solving the above problem when the search space for each polynomial coefficient is limited in the positive or negative region.

Figure 4.1 illustrate the behaviors of our polynomial debug rates $d(t_i, \hat{\boldsymbol{\alpha}})$ with degree $m = 1, 2, \ldots, 6$ in group data TIDS1 (see Table 1.1 (ii) TIDS14) in Case I. It is seen that all the software debug rates are increasing in time. On one hand, in Figure 4.2, we plot the behaviors of software debug rates with TIDS1 in Case II. As the polynomial degree m increases, the polynomial software debug rates fluctuate and can represent much more complex behaviors. It is possible to represent the non-increasing behaviors of software debug rate by relaxing the assumption of $\boldsymbol{\alpha} \in \mathbb{R}_+^{m+1}$ and to increase the log-likelihood function as well. Note that our purpose here is not to compare Case I with Case II, because Case I is involved as a special case of Case II. We aim to investigate the estimation effect between Case I and Case II, and compare our type-I and type-II local polynomial NHPP-based SRMs with the existing parametric NHPP-based SRMs.

4.4 Numerical Experiments

In numerical experiments, we use 8 software fault-detection group data sets (TIDS1∼TIDS8) (see Table 1.1 (ii) DS14∼DS21). All data sets were observed in actual mission-critical software development projects.

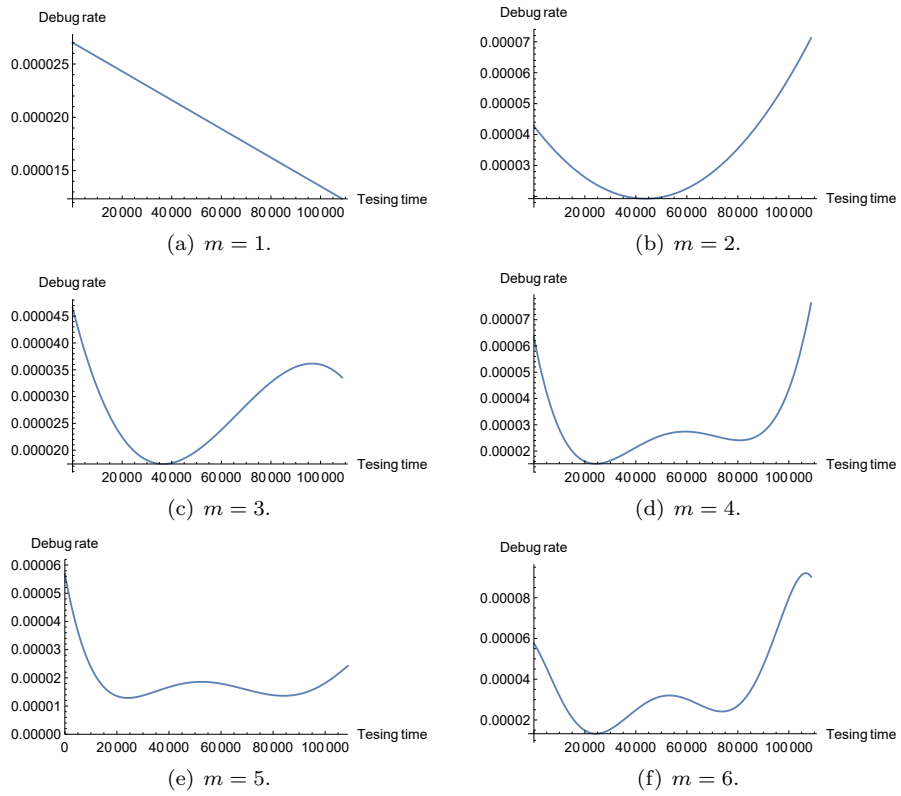


Figure 4.1: Behavior of software debug rates with TIDS1 in Case II.

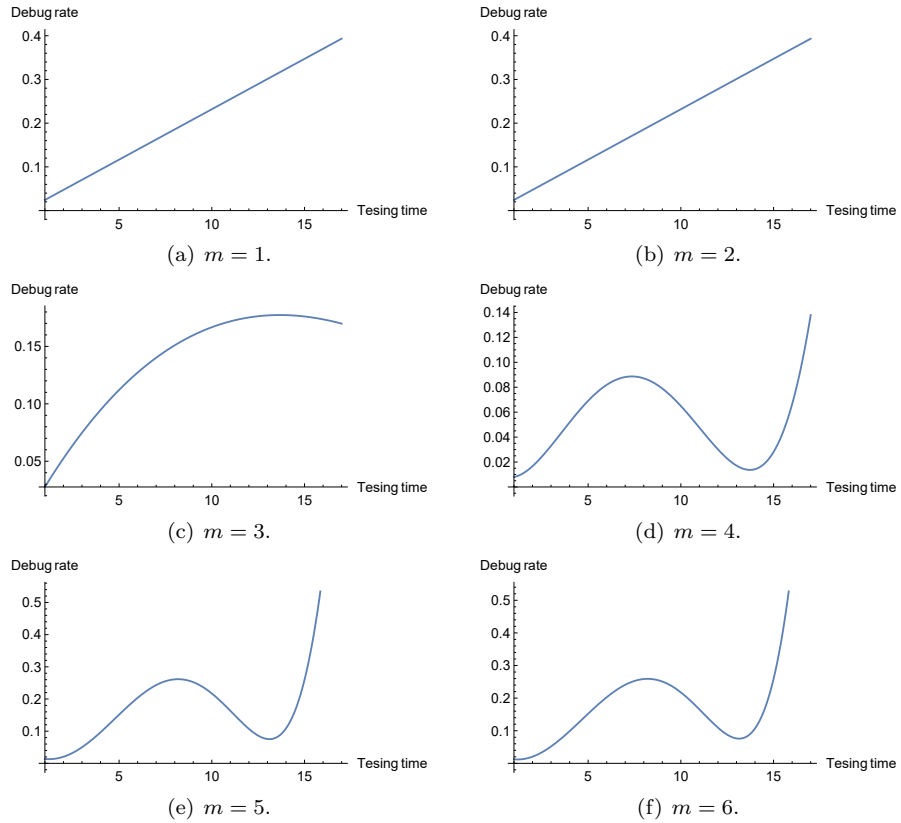
4.4.1 Goodness-of-fit Performance

To investigate the goodness-of-fit performances quantitatively, we calculate the ML estimates of the model parameters, $\hat{\boldsymbol{\theta}}$ or $\hat{\boldsymbol{\alpha}}$, for our type-I local polynomial NHPP-based SRM and type-II local polynomial NHPP-based SRM by

$$\begin{aligned} &\text{argmax}_{\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}} \ln \mathcal{L}(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}; \quad t_i, i = 1, \cdots, k) \\ &\text{s.t. } d(t_i; \hat{\boldsymbol{\alpha}}) \geq 0, i = 1, ..., k. \end{aligned} \tag{4.6}$$

AIC and MSE are used as measures for goodness-of-fit. The smaller AIC/MSE is the better SRM in terms of the goodness-of-fit to the underlying fault count data. Algorithm 1 shows an optimization procedure to find the best polynomial degree in our local polynomial NHPP-based SRMs in each data set.

Figure 4.3 plots the behavior of the mean value functions of our local polynomial and the existing NHPP-based SRMs with the group data set TIDS1. The best SRMs with minimum AIC were selected from the type-I and type-



(a) $m = 1$. (b) $m = 2$. (c) $m = 3$. (d) $m = 4$. (e) $m = 5$. (f) $m = 6$.

Figure 4.2: Behavior of software debug rates with TIDS1 in Case II.

II local polynomial NHPP-based SRMs in all combinations of $m = 1, 2, \ldots, 6$ for both Case I and Case II, and compared with the best existing type-I and type-II NHPP-based SRMs in terms of minimization of AIC. At first look, both the type-I and type-II local polynomial NHPP-based SRMs could show more accurate estimations close to actual software fault counts. More specifically, in Table 4.2, we present the AICs/MSEs of our local polynomial NHPP-based SRMs, where the polynomial degree changed from $m = 1$ to $m = 6$, and the best NHPP-based SRMs were determined with the minimum AIC. First, we notice that as the polynomial degree increases, the number of free parameters also increases and that our local polynomial NHPP-based SRMs with high-degree of polynomials could not always lead to the smaller AIC results. In fact, we examined the AIC values with $m = 7, 8, \ldots$ in our preliminary experiments, and observed that $m = 6$ is enough as the maximum polynomial degree from the viewpoint of minimization of AIC.

Table 4.2: Goodness-of-fit results.

	Polynomial NHPP-based SRMs								**Existing NHPP-based SRMs**			
	Type-I				**Type-II**				**Type-I**		**Type-II**	
	Case I		**Case II**		**Case I**		**Case II**					
	AIC	MSE	AIC	MSE	AIC	MSE	AIC	MSE	AIC	MSE	AIC	MSE
TIDS1	76.77 ($m=1$)	0.67	72.38 ($m=5$)	0.26	88.24 ($m=1$)	1.47	**71.57** (**$m=5$**)	0.32	73.05 (tlogist)	0.49	86.94 (Cox-Lewis)	1.23
TIDS2	64.97 ($m=1$)	0.46	63.09 ($m=3$)	0.39	69.51 ($m=1$)	1.40	63.34 ($m=1$)	0.53	**61.69** **(lxvmax)**	0.48	62.67 (Cox-Lewis)	0.47
TIDS3	87.72 ($m=1$)	0.65	87.48 ($m=4$)	0.33	113.79 ($m=1$)	3.87	87.89 ($m=1$)	1.04	**87.27** **(tnorm)**	0.57	91.92 (Cox-Lewis)	1.28
TIDS4	51.58 ($m=2$)	0.36	**50.64** (**$m=2$**)	0.27	61.30 ($m=1$)	2.12	51.79 ($m=5$)	0.40	51.05 (tlogist)	0.41	63.56 (Cox-Lewis)	1.51
TIDS5	31.47 ($m=1$)	0.10	31.47 ($m=1$)	0.10	29.95 ($m=1$)	0.12	**29.90** (**$m=1$**)	0.10	29.90 (exp)	0.10	29.91 (Cox-Lewis)	0.10
TIDS6	114.18 ($m=5$)	0.96	110.51 ($m=3$)	0.64	129.27 ($m=1$)	2.40	112.19 ($m=1$)	1.13	**108.83** **(lxvmax)**	1.06	112.38 (Cox-Lewis)	1.00
TIDS7	127.61 ($m=3$)	0.26	125.35 ($m=3$)	0.24	142.22 ($m=1$)	0.90	142.01 ($m=1$)	0.90	**123.27** **(txvmin)**	0.25	141.13 (Duane)	0.96
TIDS8	129.94 ($m=2$)	0.76	127.13 ($m=2$)	0.71	175.75 ($m=1$)	1.72	172.17 ($m=1$)	1.76	**117.47** **(llogist)**	0.53	174.17 (Duane)	1.87

Algorithm 1 Maximum likelihood estimation for local polynomial NHPP-based SRMs in estimation phase.

Input: $\{\mathbf{I} = (t_1, n_1), \cdots, (t_k, n_k)$, where each $t_i \in \mathbb{R}^+$ and $n_i \in \mathbb{R}^+\}$ k: testing time

Output: Parameter estimates $\hat{\boldsymbol{\alpha}} = (\hat{\mu_0}, \hat{\mu_1}, \ldots, \hat{\mu_m})$ for $m \in \mathbb{R}^+$, AIC

Parameter initialise $\boldsymbol{\alpha} = (\mu_0, \mu_1, \ldots, \mu_m)$ for $m \in \mathbb{R}^+$
$\mu_0 \in \mathbb{R}^+$, $(\mu_1, \ldots, \mu_m) \in \mathbb{R}^{m-1}$
for $i \leftarrow 1$ **to** k **do**
  $\boldsymbol{\alpha} \leftarrow \text{argmax}_{\boldsymbol{\alpha}} \text{lnL}(\boldsymbol{\alpha})$
  Compute $d(t_i;\ \boldsymbol{\alpha})$ by (9)
  if $d(t_i;\ \boldsymbol{\alpha}) > 0$ **then**
    $\hat{\boldsymbol{\alpha}} \leftarrow \boldsymbol{\alpha}$
    Compute **AIC** by (15)
    else
      for $j \leftarrow 1$ **to** m **do**
        Set the parameter range: $\mu_j \in \mathbb{R}_-$ or $\mu_j \in \mathbb{R}_+$
        Parameter combinations labeled from 1 to 2^m
        for $s \leftarrow 1$ **to** 2^j **do**
          $\boldsymbol{\alpha}_s \leftarrow \text{argmax}_{\boldsymbol{\alpha}_s} \text{lnL}(\boldsymbol{\alpha}_s)$
          Compute $d(t_i;\ \boldsymbol{\alpha}_s)_s$ by (9)
          Compute $\mathbf{AIC}_s$ by (15)
          if $\boldsymbol{AIC}_s$ *has minimum value and* $d(t_i;\ \boldsymbol{\alpha})_s > 0$ **then**
            $\hat{\boldsymbol{\alpha}} \leftarrow \boldsymbol{\alpha}_s$
            AIC $\leftarrow \mathbf{AIC}_s$
          end
        end
      end
    end
  end
end

By comparing our local polynomial NHPP-based SRMs with the existing NHPP-based SRMs in Table 4.2, it can be seen that the type-I local polynomial NHPP-based SRM in Case II could provide the smaller AIC in only TIDS4, but at the same time, it could outperform the other NHPP-based SRMs from the viewpoint of MSE in 6 cases (TIDS1 $\sim$ TIDS4, TIDS5, and TIDS6). On the other hand, the type-II local polynomial NHPP-based SRM in Case II could provide the smaller AIC in TIDS1 and TIDS5. Though the existing type-I NHPP-based SRM guaranteed the smaller AIC in a total of 5 cases, by checking Equations (4.4) and (4.5) in addition to Table 4.1, it is immediately obvious that the number of free parameters of our type-I and type-II local polynomial NHPP-based SRMs is consistent with almost all NHPP-based NHPPs (except type-I exp SRM) when $m = 2$ and $m = 3$, respectively. In other words, for each

subsequent increase in degree m, the difference in AIC between the local polynomial NHPP-based SRMs and the existing type-I NHPP-based SRMs increases by 2. On one hand, MSE exhibits the vertical distance between the estimated mean value function and the cumulative number of software fault counts, so we understood that the MSE is a more visual criterion in this comparative study.

We also notice that as the special case of our local polynomial NHPP-based SRM in Case-II, the one in Case-I could not show the better goodness-of-fit performance in any data set. It illustrates that in actual software testing, the software debug rate will not be monotonically increasing, but will fluctuate according to the difficulty of software fault fixing.

4.4.2 Predictive Performances

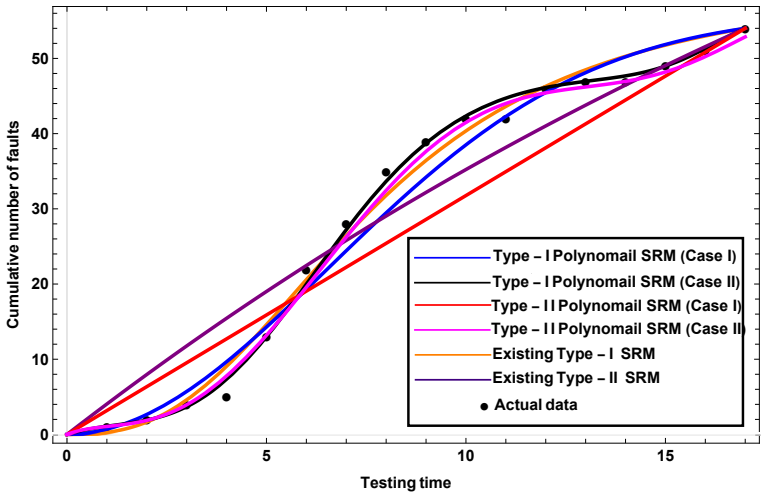
It is worth mentioning that the better goodness-of-fit to the past observation does not always lead to the better performance for future prediction. Since assessing the quantitative software reliability is equivalent to predicting the fault-free probability during a future testing/operational period, it is important to investigate the predictive performance of the NHPP-based SRMs with local polynomial software debug rate. When k and n_k, software fault count data, are available, and the prediction length is given by l $(= 1, 2, \cdots)$, we utilize the PMSE:

$$\text{PMSE} = \frac{\sqrt{\sum_{i=k+1}^{k+l}\{n_i - M(t_i; \hat{\boldsymbol{\alpha}} \text{ or } \hat{\boldsymbol{\theta}})\}^2}}{l} \tag{4.7}$$

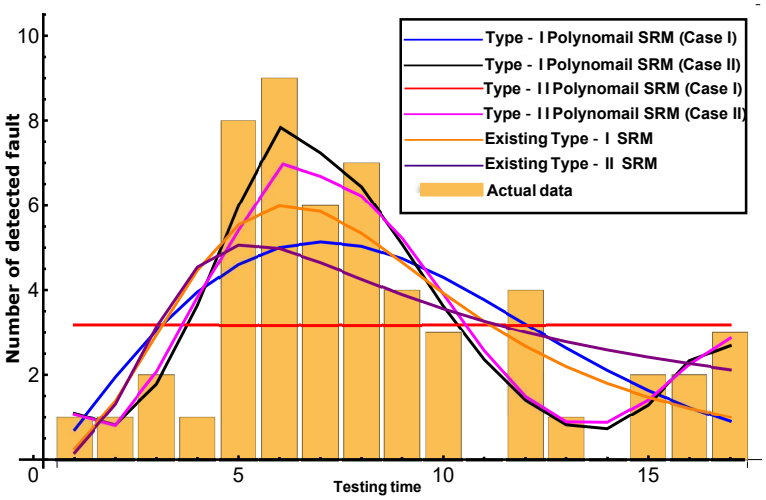
for the group data, where $\hat{\boldsymbol{\theta}}$ or $\hat{\boldsymbol{\alpha}}$ is the ML estimate obtained at t_k with constraint $d(t_i, \boldsymbol{\alpha}) \geq 0$ $(i = 1, 2, \ldots, k, \ldots, k+l)$. It is worth noting that in the prediction phase at software testing time t_k, the maximum likelihood estimates must also satisfy $d(t_i, \boldsymbol{\alpha}) \geq 0$ $(i = k+1, \ldots, k+l)$ additionally, because the future prediction has to guarantee the non-negativeness of $d(t_i, \boldsymbol{\alpha})$. In this sense, the maximum likelihood estimation in the prediction phase is modified as

$$\begin{aligned} &\text{argmax}_{\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}} \ln \mathcal{L}(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha};\ t_i, i = 1, \cdots, k) \\ &\text{s.t. } d(t_i; \boldsymbol{\alpha}) \geq 0, i = 1, ..., k, k+1, \cdots, k+l, \end{aligned} \tag{4.8}$$

which is slightly different from Equation (4.6) We give Algorithm 2 as the pseudo-code for the maximum likelihood estimation in the prediction phase of model parameters.



(a) Cumulative Number of Detected Fault Counts during testing time period.



(b) Number of Detected Fault Counts during calendar-time-based intervals.

Figure 4.3: Behavior of mean value functions in TIDS1.

Algorithm 2 Maximum likelihood estimation for local polynomial NHPP-based SRMs in prediction phase.

Input: $\{I = (t_1, n_1), \cdots, (t_k, n_k), \cdots, (t_{k+l}, n_{k+l})$, where each $t_i \in \mathbb{R}^+$ and $n_i \in \mathbb{R}^+\}$ k: testing time before observation point l: prediction time length

Output: Parameter estimates $\hat{\boldsymbol{\alpha}} = (\hat{\mu_0}, \hat{\mu_1}, \ldots, \hat{\mu_m})$ for $m \in \mathbb{R}^+$, PMSE

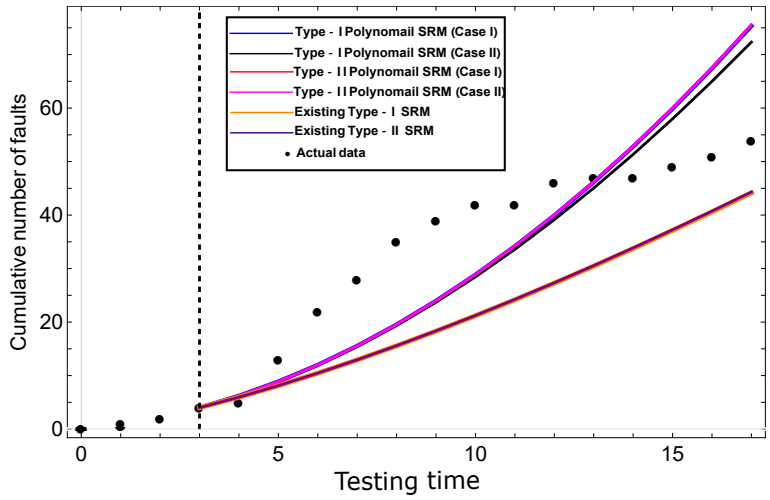
```
14 Parameter initialise α = (μ_0, μ_1, ..., μ_m) for m ∈ R^+
   μ_0 ∈ R^+, (μ_1, ..., μ_m) ∈ R^{m-1}
   for i ← 1 to k do
15   α ← argmax_α lnL(α)
      for i ← 1 to k + l do
16     Compute d(t_i; α) by (9)
        if d(t_i; α) > 0 then
17       α̂ ← α
          Compute PMSE by (18)
          else
18         for j ← 1 to m do
19           Set the parameter range: μ_j ∈ R_− or μ_j ∈ R_+
              Parameter combinations labeled from 1 to 2^m
              for s ← 1 to 2^j do
20             α_s ← argmax_{α_s} lnL(α_s)
                Compute d(t_i; α_s)_s by (9)
                Compute AIC_s by (15)
                Compute PMSE_s by (18)
                if AIC_s has minimum value and d(t_i; α)_s > 0 then
21               α̂ ← α_s
                  PMSE ← PMSE_s
22             end
23           end
24         end
25       end
26     end
27   end
28 end
```

We set the observation point k at 20%, 50% and 80% points of the whole time series data. That is, we predict the future behavior of software fault counts at $t_{k+1}, t_{k+2}, \ldots, t_{k+l}$ from the training data; $t_1, t_2, \ldots, t_k$ for the group data. Figure 4.4 shows the prediction results for the cumulative number of software faults in TIDS1 for our type-I and type-II local polynomial NHPP-based SRMs, and the existing type-I and type-II NHPP-based SRMs. As we emphasized in Chapter 3, the mean value function of type-I NHPP is bound while that of type-II NHPP is unbound, so that all type-II NHPP-based SRMs are divergent while the type-I NHPP-based SRMs are convergent, in Figure 4.4 (a), (b) and (c).

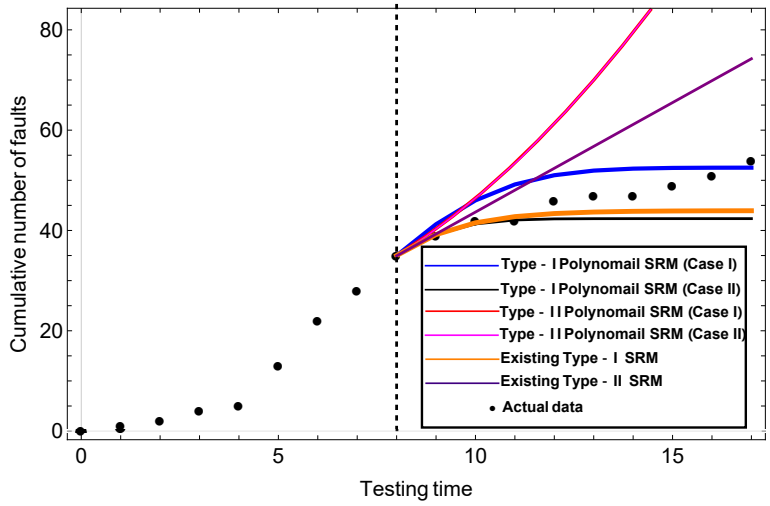
The long-term prediction at the 20% observation point is not accurate enough, but the type-II NHPP-based SRMs gave slightly higher prediction accuracy. In contrast, the type-I NHPP-based SRMs performed better in the medium- and short-term predictions. Tables 4.3, 4.4 and 4.5 present the prediction results at each observation point based on the minimum PMSE in group data sets, where we select the best SRM with the smallest PMSE from our type-I and type-II local polynomial NHPP-based SRMs with $m = 1, 2, \ldots, 6$ in both Case I and Case II, and the existing type-I/type-II NHPP-based SRMs. It is seen that when the testing phase is early (20%), our type-I local polynomial SRM in Case II could guarantee the smaller PMSE than the existing NHPP-based SRMs in TIDS1 and TIDS8, and our type-II local polynomial SRM could provide the smaller PMSE than the existing NHPP-based SRMs in 2 out of 8 cases (TIDS3 and TIDS4). When the testing phase is middle (50%), the type-I local polynomial SRM in Case I tended to give the better predictive performance in 5 cases of data sets (TIDS1, TIDS3, TIDS4, TIDS7 and TIDS8), and the type-II local polynomial SRM in Case I could provide the smaller PMSE in TIDS6. When the testing phase is later (80%), our type-I local polynomial SRM outperformed the existing NHPP-based SRMs in 4 cases of data sets (TIDS4, TIDS6, TIDS7 and TIDS8), and the type-II local polynomial NHPP-based SRM in Case II could provide the smaller PMSE in only TIDS2.

It can be noticed that our best local polynomial SRMs in Case I and Case II show the exactly same best degree and PMSE value in many data sets at each observation point. It indicates that the minimum PMSEs are provided by the local polynomial NHPP-based SRM with $\hat{\boldsymbol{\alpha}} = (\mu_0, \mu_1, \ldots, \mu_m) \in \mathbb{R}_+^{m+1}$ in both Case I and Case II, even if we consider all the parameter combinations with $\mu_0 \geq 0$ and $(\mu_1, \ldots, \mu_m) \in \mathbb{R}^m$ in the parameter estimation in Case II. This again supports the fact that Case I is involved as a special case of Case II in the local polynomial NHPP-based SRMs, as we mentioned in Section 4.3.

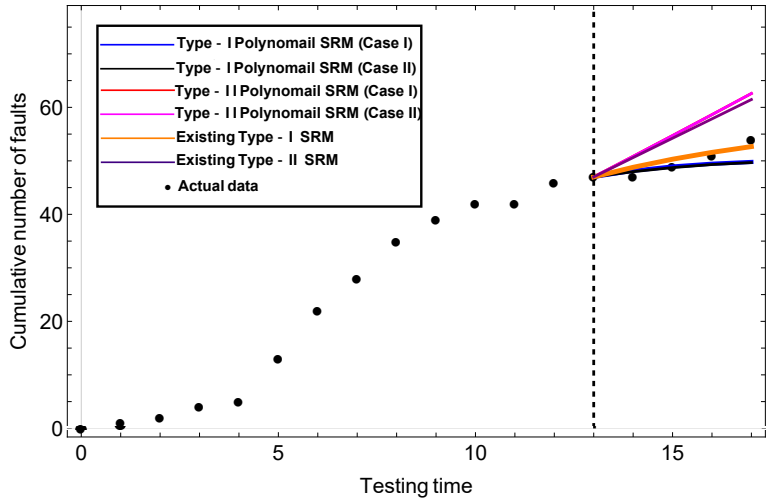
The lesson learned is that both type-II local polynomial NHPP-based SRM and type-I local polynomial NHPP-based SRM are quite competitive with the existing NHPP-based SRMs in prediction for the unknown future pattern on software fault detection in the early and later software testing phases, respectively. On one hand, our type-I local polynomial NHPP-based SRM has a high



(a) 20% observation point.



(b) 50% observation point.



(c) 80% observation point.

Figure 4.4: Predictive behavior of cumulative number of software faults in TIDS1.

potential to provide more accurate predictions of the number of residual software faults in the middle of the software testing phase. Although quite a few of the best PMSEs are guaranteed in the local polynomial NHPP-based SRMs with $m = 1$, it is not difficult to observe that the PMSEs for some local polynomial NHPP-based SRMs which outperformed the existing NHPP-based SRMs are given by the higher polynomial degree such as $m = 3 \sim 6$, through the comparison with the existing NHPP-based SRMs, in 20 % observation point with TIDS4, 50 % observation point with TIDS6 and TIDS8, and 80 % observation point with TIDS7.

Table 4.3: Predictive results (best model selection based on minimum PMSE at 20% observation point).

	Polynomial NHPP-based SRMs								Existing NHPP-based SRMs			
	Type-I				Type-II				Type-I		Type-II	
	Case I		Case II		Case I		Case II					
	Best Degree	PMSE	Best Degree	PMSE	Best Degree	PMSE	Best Degree	PMSE	Best SRM	PMSE	Best SRM	PMSE
TIDS1	m=1	2.89	**m=1**	**2.74**	m=1	2.91	m=1	2.91	gamma	3.71	Duane	3.82
TIDS2	m=4	5.45	m=5	1.96	m=1	82.25	m=3	56.55	**lxvmax**	**1.44**	Musa-Okumoto	8.72
TIDS3	m=1	12.60	m=4	7.13	**m=1**	**4.41**	**m=1**	**4.41**	gamma	6.74	Duane	6.97
TIDS4	m=6	10.58	m=4	4.86	**m=6**	**3.43**	**m=6**	**3.43**	exp	3.44	Cox-Lewis	3.60
TIDS5	m=1	5.29	m=1	5.29	m=1	5.30	m=1	5.30	**pareto**	**0.43**	Musa-Okumoto	0.45
TIDS6	m=1	5.19	m=6	3.01	m=1	12.31	m=3	12.31	**tlogist**	**2.34**	Musa-Okumoto	7.06
TIDS7	m=1	6.54	m=1	6.54	m=1	5.58	m=1	5.58	**exp**	**3.65**	**Musa-Okumoto**	**3.65**
TIDS8	**m=1**	**3.13**	**m=1**	**3.13**	m=1	3.50	m=1	3.50	txvmin	4.03	Musa-Okumoto	7.34

Table 4.4: Predictive results (best model selection based on minimum PMSE at 50% observation point).

	Polynomial NHPP-based SRMs								Existing NHPP-based SRMs			
	Type-I				Type-II				Type-I		Type-II	
	Case I		Case II		Case I		Case II					
	Best Degree	PMSE	Best Degree	PMSE	Best Degree	PMSE	Best Degree	PMSE	Best SRM	PMSE	Best SRM	PMSE
TIDS1	**m=2**	**1.36**	m=2	1.86	m=1	17.51	m=1	17.51	tlogsit	3.77	Musa-Okumoto	4.18
TIDS2	m=1	3.20	m=3	3.26	m=1	1.84	m=2	1.84	txvmin	1.84	**Duane**	**0.68**
TIDS3	**m=3**	**2.08**	m=2	3.36	m=1	14.02	m=1	14.02	lxvmax	6.83	Musa-Okumoto	10.26
TIDS4	**m=5**	**1.42**	m=2	2.29	m=1	11.72	m=1	11.72	exp	3.52	Musa-Okumoto	3.80
TIDS5	m=4	0.48	m=1	0.51	m=1	1.00	m=4	0.99	**exp**	**0.19**	Musa-Okumoto	0.21
TIDS6	m=4	5.53	m=1	5.62	**m=6**	**1.52**	m=1	1.53	pareto	5.50	Cox-Lewis	1.60
TIDS7	**m=6**	**0.28**	m=2	1.10	m=2	3.03	m=2	1.10	lxvmax	1.10	Duane	1.89
TIDS8	**m=6**	**0.92**	m=4	5.17	m=1	15.45	m=1	15.45	txvmin	1.31	Musa-Okumoto	4.61

Table 4.5: Predictive results (best model selection based on minimum PMSE at 80% observation point).

	Polynomial NHPP-based SRMs								Existing NHPP-based SRMs			
	Type-I				Type-II				Type-I		Type-II	
	Case I		Case II		Case I		Case II					
	Best Degree	PMSE	Best Degree	PMSE	Best Degree	PMSE	Best Degree	PMSE	Best SRM	PMSE	Best SRM	PMSE
TIDS1	m=1	0.93	m=4	0.95	m=1	2.68	m=1	2.68	**lnorm**	**0.53**	Musa-Okumoto	2.68
TIDS2	m=1	0.30	m=1	0.30	m=1	1.52	**m=1**	**0.29**	exp	0.30	Musa-Okumoto	0.42
TIDS3	m=1	0.37	m=1	0.37	m=1	7.40	m=5	0.28	**tnorm**	**0.23**	Musa-Okumoto	3.67
TIDS4	**m=2**	**0.17**	m=4	0.78	m=1	5.74	m=4	4.85	tnorm	0.59	Musa-Okumoto	5.30
TIDS5	m=1	0.64	m=2	0.33	m=1	0.20	m=6	0.32	tnorm	0.21	**Duane**	**0.19**
TIDS6	m=1	1.23	**m=4**	**0.54**	m=1	3.25	m=1	1.41	lnorm	0.74	Musa-Okumoto	1.34
TIDS7	m=5	0.25	**m=5**	**0.22**	m=1	4.76	m=3	1.56	txvmin	0.82	Musa-Okumoto	2.52
TIDS8	**m=1**	**0.16**	**m=1**	**0.16**	m=1	5.02	m=1	5.02	lxvmax	0.33	Musa-Okumoto	2.10

Note that there is no uniquely best NHPP-based SRM in Tables 4.3, 4.4 and 4.5 that can always guarantee the best predictive performance. In other words, it is not possible to know in advance which SRM gives the most accurate software fault prediction capability. The more common approach is to predict the future behavior of software faults using the best SRM fitted to the software fault count data observed up to the observation point if no overfitting occurs. In Tables 4.6, 4.7 and 4.8 we select the local polynomial NHPP-based SRMs and the existing NHPP-based SRMs, which gave the minimum AIC, and compared their associated predictive performances. It is obvious that all NHPP-based SRMs could not guarantee the smallest AIC and PMSE simultaneously. In TIDS3 at 50% observation point, TIDS1 and TIDS3 at 80% observation point, our type-I local polynomial NHPP-based SRM and the exiting type-I NHPP-based SRMs could provide the smaller AIC and PMSE at the same time. When we focus on only the AIC, it can be found that our type-I local polynomial NHPP-based SRMs could guarantee the smaller AIC in TIDS3 at 50% observation point, TIDS4 and TIDS6 at 80% observation point. Also, our type-II local polynomial NHPP-based SRMs could guarantee the smaller AIC in TIDS1 and TIDS5 at 20% observation point, TIDS5 at 80% observation point. In this realistic scenario that the best prediction model is unknown at the observation point, we can clearly find that our local polynomial NHPP-based SRMs could guarantee the smaller PMSE in at least half of the cases, regardless of the observation point, where the PMSEs of TIDS1 in Table 4.6, TIDS1, TIDS3 and TIDS4 in Table 4.7 and TIDS4 in Table 4.8 have the more significant differences when compared with the existing NHPP-based SRMs.

Table 4.6: Predictive results (best model selection based on minimum AIC) at 20% observation point.

	Polynomial NHPP-based SRMs								Existing NHPP-based SRMs			
	Type-I				Type-II				Type-I		Type-II	
	Case I		Case II		Case I		Case II					
	AIC (Best Degree)	PMSE	AIC (Best Degree)	PMSE	AIC (Best Degree)	PMSE	AIC (Best Degree)	PMSE	AIC (Best SRM)	PMSE	AIC (Best SRM)	PMSE
TIDS1	12.75 (m=1)	2.89	12.75 (m=1)	**2.74**	**10.75 (m=1)**	2.91	**10.75 (m=1)**	2.91	11.09 (exp)	6.24	10.76 (Cox-Lewis)	8.87
TIDS2	13.58 (m=1)	82.13	13.58 (m=1)	82.13	12.58 (m=1)	82.25	12.58 (m=1)	82.25	12.87 (lxvmax)	**1.44**	**10.86 (Musa-Okumoto)**	8.72
TIDS3	20.14 (m=1)	12.60	19.83 (m=2)	16.20	19.59 (m=1)	**4.41**	17.83 (m=2)	956.164	18.44 (exp)	10.15	**17.20 (Duane)**	6.97
TIDS4	12.65 (m=1)	10.62	12.65 (m=1)	10.62	11.67 (m=1)	**3.43**	11.67 (m=1)	**3.43**	11.67 (exp)	3.44	**10.65 (Duane)**	8.02
TIDS5	8.58 (m=1)	5.29	8.58 (m=1)	5.29	**6.58 (m=1)**	5.30	**6.58 (m=1)**	5.30	7.39 (exp)	**0.43**	9.39 (Musa-Okumoto)	**0.43**
TIDS6	23.83 (m=2)	5.37	23.83 (m=2)	5.37	33.16 (m=1)	12.31	33.16 (m=1)	12.31	**20.66 (lnorm)**	**5.15**	32.76 (Duane)	17.00
TIDS7	23.49 (m=2)	6.60	21.82 (m=2)	6.60	25.33 (m=1)	5.58	25.33 (m=1)	5.58	**16.96 (txvmin)**	6.60	27.58 (Musa-Okumoto)	**3.65**
TIDS8	13.36 (m=1)	**3.13**	13.36 (m=1)	**3.13**	13.36 (m=1)	3.50	13.36 (m=1)	3.50	**8.61 (txvmin)**	4.03	15.78 (Musa-Okumoto)	7.34

Table 4.7: Predictive results (best model selection based on minimum AIC) at 50% observation point.

	Polynomial NHPP-based SRMs								Existing NHPP-based SRMs			
	Type-I				Type-II				Type-I		Type-II	
	Case I		Case II		Case I		Case II					
	AIC (Best Degree)	PMSE	AIC (Best Degree)	PMSE	AIC (Best Degree)	PMSE	AIC (Best Degree)	PMSE	AIC (Best SRM)	PMSE	AIC (Best SRM)	PMSE
TIDS1	37.10 (m=1)	17.38	36.25 (m=2)	**1.86**	35.10 (m=1)	17.51	35.10 (m=1)	17.51	36.85 (lxvmin)	4.35	**35.09 (Duane)**	19.02
TIDS2	35.16 (m=1)	3.20	35.16 (m=1)	3.20	40.57 (m=1)	1.84	35.75 (m=3)	56.55	**31.95 (lxvmax)**	2.63	38.97 (Musa-Okumoto)	**0.92**
TIDS3	50.61 (m=1)	14.02	**48.09 (m=2)**	**3.36**	48.61 (m=1)	14.02	48.61 (m=1)	14.02	49.31 (exp)	9.32	48.52 (Cox-Lewis)	15.58
TIDS4	31.40 (m=1)	11.72	30.40 (m=2)	**2.29**	29.40 (m=1)	11.72	29.40 (m=2)	28.76	30.56 (tlogist)	22.97	**28.56 (Cox-Lewis)**	23.10
TIDS5	18.77 (m=1)	0.51	18.77 (m=1)	0.51	17.12 (m=1)	1.00	17.12 (m=1)	1.00	17.37 (exp)	**0.19**	**16.88 (Duane)**	0.80
TIDS6	55.32 (m=1)	5.62	44.64 (m=3)	5.88	74.12 (m=1)	**1.53**	52.18 (m=2)	16.39	**40.52 (lxvmax)**	5.63	65.62 (Duane)	2.25
TIDS7	73.02 (m=1)	1.84	73.02 (m=1)	1.84	71.03 (m=1)	3.03	71.03 (m=1)	3.03	72.39 (lxvmax)	**1.10**	**70.52 (Duane)**	1.89
TIDS8	74.41 (m=3)	32.27	71.50 (m=2)	26.70	72.49 (m=3)	71.23	72.49 (m=3)	71.23	72.39 (txvmin)	**1.31**	**68.49 (Duane)**	70.20

Table 4.8: Predictive results (best model selection based on minimum AIC) at 80% observation point.

	Polynomial NHPP-based SRMs								Existing NHPP-based SRMs			
	Type-I				Type-II				Type-I		Type-II	
	Case I		Case II		Case I		Case II					
	AIC (Best Degree)	PMSE	AIC (Best Degree)	PMSE	AIC (Best Degree)	PMSE	AIC (Best Degree)	PMSE	AIC (Best SRM)	PMSE	AIC (Best SRM)	PMSE
TIDS1	58.19 (m=2)	1.53	58.19 (m=2)	1.53	71.87 (m=1)	2.68	59.34 (m=4)	20.20	**56.86 (lxvmin)**	**1.39**	69.77 (Duane)	3.52
TIDS2	55.85 (m=1)	0.30	53.89 (m=3)	0.90	57.08 (m=1)	1.52	53.91 (m=1)	**0.29**	**52.52 (lxvmax)**	0.42	55.28 (Duane)	0.90
TIDS3	76.07 (m=2)	0.93	75.39 (m=2)	1.50	83.42 (m=1)	7.40	76.39 (m=4)	12.90	**75.29 (txvmin)**	**0.89**	79.12 (Duane)	1.34
TIDS4	44.53 (m=2)	**0.17**	**41.27 (m=2)**	0.92	46.52 (m=1)	5.74	43.22 (m=3)	4.85	42.54 (txvmin)	0.83	46.28 (Duane)	5.37
TIDS5	25.19 (m=1)	0.64	25.19 (m=1)	0.64	26.33 (m=1)	**0.20**	**24.25 (m=1)**	0.33	24.27 (exp)	0.29	24.27 (Duane)	**0.20**
TIDS6	101.34 (m=1)	1.23	**74.79 (m=5)**	18.28	104.60 (m=1)	3.25	74.96 (m=5)	73.17	96.18 (lxvmax)	0.94	99.16 (Musa-Okumote)	1.34
TIDS7	113.40 (m=1)	2.81	113.40 (m=1)	2.81	112.50 (m=1)	4.76	112.50 (m=1)	4.76	112.84 (txvmin)	**0.82**	**111.46 (Musa-Okumote)**	2.52
TIDS8	114.02 (m=2)	**0.79**	114.02 (m=6)	0.87	151.78 (m=1)	5.02	151.78 (m=1)	5.02	**100.33 (tlogist)**	0.86	147.93 (Duane)	3.63

Chapter 5

Proportional Intensity-based SRMs

This chapter focuses on the so-called proportional intensity-based software reliability models (PI-SRMs), which are extensions of the common NHPP-based SRMs, and describe the probabilistic behavior of software fault-detection process by incorporating the time-dependent software metrics data observed in the development process. The PI-SRM is proposed by Rinsaka et al. in the paper "PISRAT: Proportional Intensity-Based Software Reliability Assessment Tool" in 2006. Specifically, we generalize this seminal model by introducing eleven well-known fault-detection distributions.

5.1 Preliminary

The existing common NHPP-based SRMs are generally characterized by the mean value functions or the c.d.f.s of software fault-detection time. Hence, they can quantitatively represent the typical software reliability growth phenomena and the software debugging scenarios during the software testing phase. In other words, the above approach is categorized into a black-box approach, where the software fault-detection time distribution is estimated with only the fault count data and does not depend on the knowledge/learning effects of the software product, test resources, and the process information. It should be noted that the common NHPP-based SRMs are quite simple in software reliability measurement and fault prediction but miss out on several software development/testing metrics of data collected throughout the software development process.

In this chapter, we summarize the so-called proportional intensity-based software reliability models (PI-SRMs) by Rinsaka et al. [99], which are extensions of the common NHPP-based SRMs, As an extension of the common NHPP-based SRMs, this chapter summarizes the so-called proportional intensity-based software reliability models (PI-SRMs), and describes the probabilistic behavior of the software fault-detection process by incorporating the time-dependent software metrics data observed in the development process. In the subsequent paper, Shibata et al. [100] develop a software reliability assessment tool, PI-SRAT, to automate the parameter estimation and quantify the software reliability. We generalize the seminal PI-SRM in [99] by introducing several well-known fault-detection time distributions because the work in [99] limited a few kinds of software fault-detection time distributions. The advantage of PI-SRMs is to combine a regression formula to represent the dependence of software metrics data with a stochastic counting process for the software fault counts. Similar to the well-known software reliability assessment tool in SRATS [43], we introduce eleven parametric models (see Table 5.1) (baseline intensity functions) in the PI-SRM and comprehensively evaluate the potential performances.

5.2 Proportional Intensity Model

5.2.1 Model Description

Suppose that l types of software metrics data, $\boldsymbol{x}_k = (x_{k1}, \cdots, x_{kl})$ $(k = 1, 2, \ldots, n)$, are observed at each testing time t_k $(= 0, 1, 2, \ldots, n)$. For analytical purposes, we assume that each software metric $\boldsymbol{x}_k$ is dependent on the cumulative testing time t_k, and can be considered as a time-dependent function, denoted by $\boldsymbol{x}_k(t_k)$. In fact, this sort of parameter is referred to as a time-dependent covariate [101, 102] in statistics and has been widely investigated in the context of the Cox regression-based proportional hazard model (PHM). We define the intensity function for our PI-SRM by:

$$\lambda_x(t_k, \boldsymbol{x}_k; \boldsymbol{\theta}, \boldsymbol{\beta}) = \lambda_0(t_k; \boldsymbol{\theta}) g(\boldsymbol{x}_k; \boldsymbol{\beta}), \tag{5.1}$$

with the regression coefficients $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_l)$ and the baseline intensity $\lambda_0(t_k; \boldsymbol{\theta})$ (> 0), and the covariate function $g(\boldsymbol{x}_k; \boldsymbol{\beta})$ (> 0). When $g(\boldsymbol{x}_k; \boldsymbol{\beta}) = 1$ for any $\boldsymbol{x}_k$, the PI-SRMs are reduced to the NHPP-based SRMs with the base-

Table 5.1: The baseline intensity function of existing NHPP-based SRMs.

SRM	$\boldsymbol{\lambda(t;\theta)}$
Exponential distribution (exp)	$\lambda(t;\boldsymbol{\theta}) = \omega b e^{-bt}$
Gamma distribution (gamma)	$\lambda(t;\boldsymbol{\theta}) = \omega \frac{e^{-\frac{t}{c}}\left(\frac{t}{c}\right)^{b-1}}{c\Gamma(b)}$
Pareto distribution (pareto)	$\lambda(t;\boldsymbol{\theta}) = \frac{\omega bc\left(\frac{c}{c+t}\right)^{b-1}}{(c+t)^2}$
Truncated normal distribution (tnorm)	$\lambda(t;\boldsymbol{\theta}) = \frac{\omega e^{-\frac{(c-t)^2}{2b^2}}}{\sqrt{2\pi}b\left(1-\frac{1}{2}erfc\left(\frac{c}{\sqrt{2}b}\right)\right)}$
Log-normal distribution (lnorm)	$\lambda(t;\boldsymbol{\theta}) = \frac{\omega e^{-\frac{(c-\log(t))^2}{2b^2}}}{\sqrt{2\pi}bt}$
Truncated logistic distribution (tlogist)	$\lambda(t;\boldsymbol{\theta}) = \frac{\omega e^{-\frac{t-c}{b}}}{b\left(1-\frac{1}{e^{c/b}+1}\right)\left(e^{-\frac{t-c}{b}}+1\right)^2}$
Log-logistic distribution (llogist)	$\lambda(t;\boldsymbol{\theta}) = \frac{\omega e^{-\frac{\log(t)-c}{b}}}{bt\left(e^{-\frac{\log(t)-c}{b}}+1\right)^2}$
Truncated extreme-value maximum distribution (txvmax)	$\lambda(t;\boldsymbol{\theta}) = \frac{\omega e^{-\frac{t-c}{b}-e^{-\frac{t-c}{b}}}}{b\left(1-e^{-e^{c/b}}\right)}$
Log-extreme-value max maximum distribution (lxvmax)	$\lambda(t;\boldsymbol{\theta}) = \frac{\omega c e^{-\left(\frac{t}{b}\right)^{-c}}\left(\frac{t}{b}\right)^{-c-1}}{b}$
Truncated extreme-value minimum distribution (txvmin)	$\lambda(t;\boldsymbol{\theta}) = \frac{\omega e^{-\frac{-c-t}{b}-e^{-\frac{-c-t}{b}}+e^{c/b}}}{b}$
Log-extreme-value minimum distribution (lxvmin)	$\lambda(t;\boldsymbol{\theta}) = \frac{\omega e^{-\frac{-c-\log(t)}{b}-e^{-\frac{-c-\log(t)}{b}}}}{bt}$

line intensity $\lambda_0(t;\boldsymbol{\theta})$. Based on the idea of common Cox regression PHM, it is appropriate to assume the following exponential form for the covariate function:

$$g(\boldsymbol{x}_k;\boldsymbol{\beta}) = \exp(\boldsymbol{x}_k\boldsymbol{\beta}). \tag{5.2}$$

In the literature [101, 102, 103], the above form is widely accepted to make the analysis easy and flexible. Lawless [104] also analyzed the event count data in actual medical applications with the same exponential covariate function. Note that the time-independent covariates considered by Lawless [104] were the binary data taking 0 or 1. Rinsaka et al. [99] proposed an intuitive but reasonable model to deal with the effect of the cumulative number of software faults and the software metrics in the covariate function. Define the mean value function for the given data $(t_k, y_k, \boldsymbol{x}_k)$ $(k = 1, 2, \ldots, n)$ by:

$$\begin{aligned}
M_p(t_1;\boldsymbol{\theta},\boldsymbol{\beta}) &= \int_0^{t_1} \lambda_0(u;\boldsymbol{\theta})\exp(\boldsymbol{x}_1\boldsymbol{\beta})du, && (5.3)\\
M_p(t_2;\boldsymbol{\theta},\boldsymbol{\beta}) &= \int_{t_1}^{t_2} \lambda_0(u;\boldsymbol{\theta})\exp(\boldsymbol{x}_2\boldsymbol{\beta})du + M_p(t_1;\boldsymbol{\theta},\boldsymbol{\beta}), && (5.4)\\
&\vdots \\
M_p(t_k;\boldsymbol{\theta},\boldsymbol{\beta}) &= \sum_{i=1}^{k}\exp(\boldsymbol{x}_i\boldsymbol{\beta})\int_{t_{i-1}}^{t_i}\lambda_0(u;\boldsymbol{\theta})du \\
&= \sum_{i=1}^{k}\exp(\boldsymbol{x}_i\boldsymbol{\beta})\times\left[M_0(t_i;\boldsymbol{\theta}) - M_0(t_{i-1};\boldsymbol{\theta})\right], && (5.5)
\end{aligned}$$

where $M_0(t_i;\boldsymbol{\theta}) = \int_0^{t_i}\lambda_0(u;\boldsymbol{\theta})du$. It is seen again that the PI-SRM can be reduced to the common NHPP-based SRM when $\beta_j = 0$ for all j $(= 1, 2, \cdots, l)$. By introducing $M_p(t;\boldsymbol{\theta},\boldsymbol{\beta})$, we confirm that the monotone property of the mean value function with respect to testing time r can be guaranteed. Substituting the intensity function in Table 5.1 into the baseline intensity $\lambda_0(t;\boldsymbol{\theta})$, we obtain the eleven PI-SRMs corresponding to the NHPP-based SRMs in SRATS [43].

5.2.2 Maximum Likelihood Estimation

We also utilize the maximum likelihood estimation to estimate the parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ of PI-SRM. For the fault count data (t_k, y_k) and software metrics data $\boldsymbol{x}_k = (x_{k1}, \cdots, x_{kl})$ $(k = 1, 2, \ldots, n)$, we define the likelihood function by:

$$\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\beta}) = \prod_{k=1}^{n}\frac{\{M_p(t_k;\boldsymbol{\theta},\boldsymbol{\beta}) - M_p(t_{k-1};\boldsymbol{\theta},\boldsymbol{\beta})\}^{y_k - y_{k-1}}}{(y_k - y_{k-1})!}\exp(-M_p(t_n;\boldsymbol{\theta},\boldsymbol{\beta})), \tag{5.6}$$

so that the log-likelihood function of PI-SRM can be written as:

$$\begin{aligned}\ln \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\beta}) &= \sum_{k=1}^{n} \ln \left[M_p(t_k; \boldsymbol{\theta}, \boldsymbol{\beta}) - M_p(t_{k-1}; \boldsymbol{\theta}, \boldsymbol{\beta}) \right] (y_k - y_{k-1}) \\ &\quad - \sum_{k=1}^{n} \ln \left[(y_k - y_{k-1})! \right] - M_p(t_n; \boldsymbol{\theta}, \boldsymbol{\beta}). \end{aligned} \tag{5.7}$$

By maximizing Equation (5.7) with the Newton–Raphson method, we obtain the maximum likelihood estimates $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}})$ of PI-SRM.

5.3 Numerical Examples

In our numerical examples, four software fault count data with software metrics are used, where these data are measured in the real-time command and control system development projects [24]. We re-name these data as GDS1 $\sim$ GDS4. Details are shown in Table 1.1 (iv), in which three software metrics data: failure identification work, execution time, and computer time-failure identification, are involved in addition to the cumulative number of software faults detected at each testing time (calendar week in [24]). We quantitatively evaluate the goodness-of-fit performances of eleven PI-SRMs and evaluate the predictive performances via the above four time-dependent metrics data as the covariates. In the following discussion, we consider two patterns in dealing with software metrics. One is to input the software metrics as the cumulative $\boldsymbol{x}_k = (x_{k1}, \cdots, x_{kl})$, the other as the difference $\boldsymbol{x}_k = \left(x_{k1} - x_{(k-1)1}, \cdots, x_{kl} - x_{(k-1)l}\right)$, where l is the number of time-dependent metrics data in each data set and $k = 0, 1, 2, \cdots, n$. The main concern here is to investigate the effects of cumulative values of software metrics on the contribution to the software fault count. For instance, we examine the difference between the cumulative length of test execution time by the present testing time and the test execution time spend on the same testing time.

5.3.1 Goodness-of-fit Performances

For our PI-SRMs, we assume eleven baseline intensity functions in Table 5.1 and compare them to investigate the effects of each time-dependent software metric data on the stochastic behavior of the cumulative number of software faults detected in the testing phase. We calculate the maximum likelihood estimates $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}})$ of covariate $g(\boldsymbol{x}_k; \boldsymbol{\beta}) = \exp(\boldsymbol{x}_k \boldsymbol{\beta})$ for all combinations of software metrics

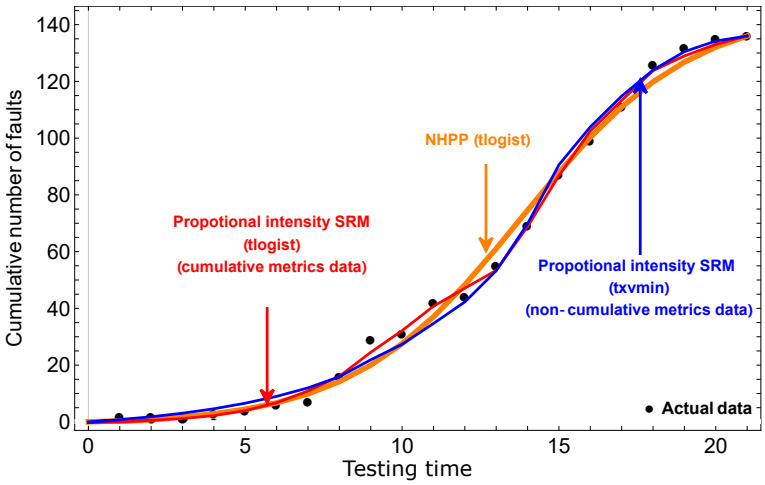


Figure 5.1: Behavior of estimated cumulative number of software faults in GDS1.

data in Table 1.1 (iv) and consider a total of 7 combinations, as shown in Table 5.2. By deriving the corresponding log likelihood function, the AIC and MSE are used to evaluate the goodness-of-fit performances of our PI-SRMs.

Table 5.2: Combination of covariates $g(\boldsymbol{x}_{kl};\boldsymbol{\beta})$.

	$\boldsymbol{g(x_{kl};\beta)(l=1,2,3)}$
Combination I	$\exp(\beta_0 + x_{k1}\beta_1)$
Combination II	$\exp(\beta_0 + x_{k2}\beta_2)$
Combination III	$\exp(\beta_0 + x_{k3}\beta_3)$
Combination IV	$\exp(\beta_0 + x_{k1}\beta_1 + x_{k2}\beta_2)$
Combination V	$\exp(\beta_0 + x_{k1}\beta_1 + x_{k3}\beta_3)$
Combination VI	$\exp(\beta_0 + x_{k2}\beta_2 + x_{k3}\beta_3)$
Combination VII	$\exp(\beta_0 + x_{k1}\beta_1 + x_{k2}\beta_2 + x_{k3}\beta_3)$
x_{k1} : Execution time, x_{k2} : Failure identification work.	
x_{k3} : Computer time-failure identification.	

In Figure 5.1, we plot the cumulative number of detected software faults in GDS1 and the estimated mean value functions in the best-fitted SRMs, where we select the best model with the minimum AIC for the common NHPP-based SRMs without software metrics (orange curve) in SRATS [43], PI-SRM with

cumulative software metrics (red curve), and PI-SRM with non-cumulative software metrics (blue curve), among eleven intensity functions. At first glance, it can be seen that the three curves exhibit similar behavior, but a closer look reveals that our PI-SRMs can show more complex behaviors than the existing NHPP-based SRMs without software metrics. Figure 5.2 illustrates the behavior of the estimated number of detected fault counts at each testing time interval in GDS1, where the same models as Figure 5.1 are used for comparison, and the orange bar-chart represents the actual number of software faults in each testing week. The result explains that our two PI-SRMs could show better goodness-of-fit performances than the existing NHPP-based SRM without software metrics and could catch up with the detailed trend on the software fault count.

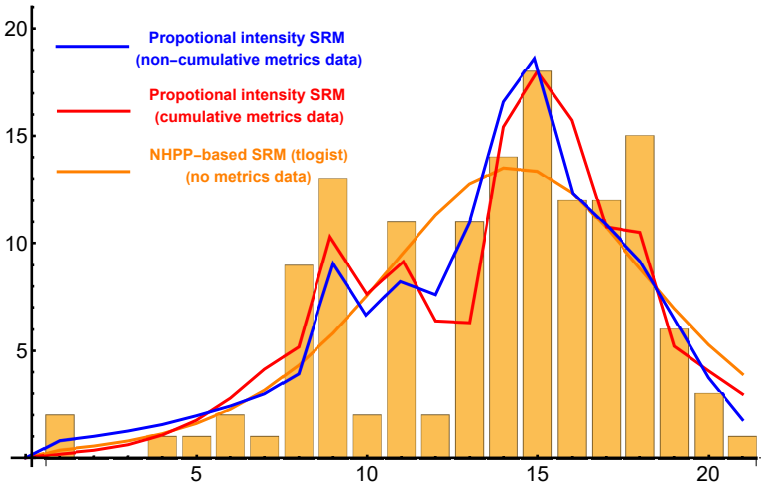


Figure 5.2: Behavior of estimated number of software faults in each time interval in GDS1.

To compare our PI-SRMs with the common NHPP-based SRMs without software metrics more precisely, we present the best AIC results for four time-dependent metrics data in Table 5.3. By comparing our two PI-SRMs with cumulative/non-cumulative metrics values, we investigate how to deal with the software metrics data in software fault data analysis. From the results in Table 5.3, it is found that our PI-SRMs are more appealing in software reliability modeling and outperform the existing NHPP-based SRMs without software metrics in terms of goodness-of-fit. In the comparison of two patterns with cumulative/non-cumulative metric data, it is seen that the non-cumulative soft-

ware metrics tend to show better fitting results except in GDS4. Note that the difference of AIC between cumulative/non-cumulative metric patterns is minimal and negligible. Therefore, our conclusion on the goodness-of-fit performance is that the PI-SRM with non-cumulative software metric data should be better. Furthermore, in Table 5.3, it is observed that both the execution time and failure identification work could contribute to the goodness-of-fit performance in the PI-SRMs. Hence, the measurement of test execution time and failure identification work can help to understand the software fault count in the testing phase more accurately and is useful to monitor the software testing progress.

Table 5.3: Goodness-of-fit performance based on AIC.

	Model	AIC	MSE	$\hat{\boldsymbol{\beta}}$
(i) Best proportional intensity model (cumulative metrics data)				
GDS1	tlogist-VI	110.114	0.470	$\hat{\beta_0} = -2.5903, \hat{\beta_2} = -0.0805, \hat{\beta_3} = 0.0277$
GDS2	tlogist-III	69.785	0.282	$\hat{\beta_0} = 1.7326, \hat{\beta_3} = 0.1406$
GDS3	txvmin-II	57.281	0.289	$\hat{\beta_0} = -3.7048, \hat{\beta_2} = 1.2197$
GDS4	exp-I	81.059	0.612	$\hat{\beta_0} = 4.6132, \hat{\beta_1} = -0.1659$
(ii) Best proportional intensity model (non-cumulative metrics data)				
GDS1	txvmin-II	109.015	0.721	$\hat{\beta_0} = 2.9503, \hat{\beta_2} = 0.0206$
GDS2	llogist-II	67.352	0.261	$\hat{\beta_0} = -0.4155, \hat{\beta_2} = 0.0447$
GDS3	gamma-II	50.696	0.221	$\hat{\beta_0} = 0.6061, \hat{\beta_2} = 1.1493$
GDS4	exp-VI	81.131	0.450	$\hat{\beta_0} = 3.8840, \hat{\beta_2} = -0.2963, \hat{\beta_3} = 0.8060$
(iii) Best SRATS (no metrics data)				
GDS1	tlogist	116.891	0.820	-
GDS2	llogist	73.053	0.501	-
GDS3	lxvmax	61.694	0.481	-
GDS4	txvmin	79.761	0.530	-

5.3.2 Predictive Performances

Next, we are concerned with investigating the predictive performances of our PI-SRMs. In each observation point n' $(1 \leq n' < n)$ when 50% or 80% of the whole data are available, we predict the future behavior of the cumulative number of software faults. To assess the predictive ability, we apply the PMSE as the predictive performance measure, where:

$$\mathrm{PMSE} = \frac{1}{n - \acute{n}} \sqrt{\sum_{k=\acute{n}+1}^{n} \left[y_k - M_p(t_k; \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}}) \right]^2}. \tag{5.8}$$

The smaller the PMSE, the better the prediction performance of the model. As expected, when we predict the number of software faults detected in the future, both the software metrics $\boldsymbol{x}_k$ $(k = 1, 2, \ldots, n)$ and the regression coefficient $\boldsymbol{\beta}$ must be estimated. The regression coefficients are available by applying the plug-in estimates (maximum likelihood estimates) with the past observation. However, the difficulty when the PI-SRMs are used arises since we have to predict the software metrics themselves in the future. In our numerical experiments, we consider the following three cases:

Case I: All the test/development metric data are completely known through the testing phase in advance, so the software testing expenditures are exactly given in the testing.

Case II: The test/development metrics data do not change from the observation point to the future.

Case III: The test/development metrics data experienced in the future are regarded as independent random variables and predictable by any statistical method.

Case I corresponds to the case where the software test plan is established and there is no confusion in the software testing phase. Case II implicitly assumes that the observation point is regarded as the release point of software because no testing effort will be spent in the operational phase. Case III would be the most plausible case in software testing. In this case, we are requested to introduce any statistical model to investigate the test/development metrics data. We employ two elementary regression methods, linear regression and exponential regression,

to predict the future software metrics data. More specifically, we assume that the metric data $\boldsymbol{x}_k$ before the observation point n' and the corresponding time point t_k have been observed with $k = 1, 2, \ldots, n'$. Next, our goal is to calculate the predictive value of the metric data $\hat{\boldsymbol{x}_k}$ between a given time interval $(t_{n'+1},\ t_n)$, by introducing the independent variable $T = \{t_{n'+1}, t_{n'+2}, ...t_n\}$ into the linear regression equation:

$$\hat{\boldsymbol{x}_k} = \delta_1 + \delta_2 t_k \tag{5.9}$$

with intercept δ_1 and coefficient δ_2, where:

$$\delta_1 = \frac{\left(\sum_{k=1}^{n'} \boldsymbol{x}_k\right)\left(\sum_{k=1}^{n'} {t_k}^2\right) - \left(\sum_{k=1}^{n'} t_k\right)\left(\sum_{k=1}^{n'} t_k \boldsymbol{x}_k\right)}{n'\left(\sum_{k=1}^{n'} {t_k}^2\right) - \left(\sum_{k=1}^{n'} t_k\right)^2} \tag{5.10}$$

and

$$\delta_2 = \frac{n'\left(\sum_{k=1}^{n'} t_k \boldsymbol{x}_k\right) - \left(\sum_{k=1}^{n'} t_k\right)\left(\sum_{k=1}^{n'} \boldsymbol{x}_k\right)}{n'\left(\sum_{k=1}^{n'} {t_k}^2\right) - \left(\sum_{k=1}^{n'} t_k\right)^2}. \tag{5.11}$$

Similar to the linear regression method, we can also obtain the predictive values of the metric data $\hat{\boldsymbol{x}_k}$ by importing variable $T = \{t_{n'+1}, t_{n'+2}, ...t_n\}$ into the exponential regression equation:

$$\hat{\boldsymbol{x}_k} = \delta_3 {\delta_4}^{t_k}, \tag{5.12}$$

where the coefficients δ_3 and δ_4 are given by:

$$\delta_3 = \exp\left(\frac{\left(\sum_{k=1}^{n'} \ln \boldsymbol{x}_k\right)\left(\sum_{k=1}^{n'} {t_k}^2\right) - \left(\sum_{k=1}^{n'} t_k\right)\left(\sum_{k=1}^{n'} t_k \ln \boldsymbol{x}_k\right)}{n'\left(\sum_{k=1}^{n'} {t_k}^2\right) - \left(\sum_{k=1}^{n'} t_k\right)^2}\right) \tag{5.13}$$

and,

$$\delta_4 = \exp\left(\frac{n'\left(\sum_{k=1}^{n'} t_k \ln \boldsymbol{x}_k\right) - \left(\sum_{k=1}^{n'} t_k\right)\left(\sum_{k=1}^{n'} \ln \boldsymbol{x}_k\right)}{n'\left(\sum_{k=1}^{n'} {t_k}^2\right) - \left(\sum_{k=1}^{n'} t_k\right)^2}\right) \tag{5.14}$$

respectively. Note that with Equations (5.13) and (5.14), it can be easily found that the exponential regression is not appropriate for making the prediction when the non-cumulative metric data in PI-SRMs are used, because the variable may take 0, and the correlation coefficient may not be calculated theoretically. Therefore, we totally consider seven patterns of estimated development/test metrics data in the future phase in the above three cases, and investigate the predictive performances of our PI-SRMs.

Figures 5.3 and 5.4 depict the prediction results of the cumulative number of software faults in GDS1 at 50% observation and 80% observation, respectively. It is not difficult to find that our two PI-SRMs could show a completely different predictive trend than the common NHPP-based SRMs. However, we can recognize that the closer increasing trend to the underlying software fault count data, no matter whether the prediction length is long or short, especially in the testing phase after 50% and 80% observation points. The quantitative comparison in terms of predictive performance is investigated in Tables 5.4 and 5.5, where we present the PMSE in four data sets at 50% observation point and 80 % observation point, respectively. Here we select the best SRMs with the smallest PMSE in PI-SRMs with cumulative/non-cumulative software metric data in CASE I, CASE II, and CASE III, and the existing NHPP-based SRMs. From these results, it is immediate to see that our PI-SRMs could still outperform the existing NHPP-based SRMs in all the data sets. We also find that utilizing the estimated metrics data in Case II, i.e., when the test/development metrics data do not change in the future, tends to give better predictive performances than the other two cases in many cases (GDS1 50%, GDS2 50%, GDS3 50%, GDS4 50%, and GDS4 80%). So in 5 out of 8 (GDS2 50%, GDS4 50%, GDS2 80%, GDS3 80%, and GDS4 80%); our PI-SRMs with non-cumulative metric data could provide the minimum PMSE. More specifically, Combination II of software metrics in Table 2.2 gives the minimum PMSEs in GDS1 80%, GDS3 80%, and GDS4 80% data sets with non-cumulative software metric data and GDS1 50%, GDS2 50% with cumulative software metric data, respectively. The remaining three minimum PMSEs were given in the PI-SRMs with Combinations V, VI, and VII in Table 5.3. Finally, by carefully checking the prediction results in Tables 5.4 and 5.5, we conclude that the failure identification work is the most important development metric in prediction and leads to improving the software fault prediction accurately.

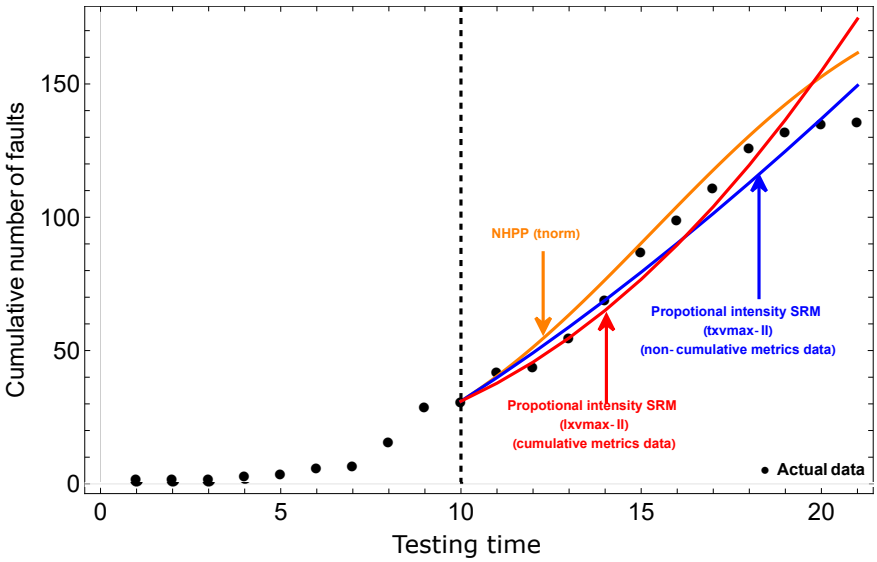


Figure 5.3: Behavior of the predicted cumulative number of software faults with PI-SRMs and common NHPP-based SRM in GDS1 (50% observation point).

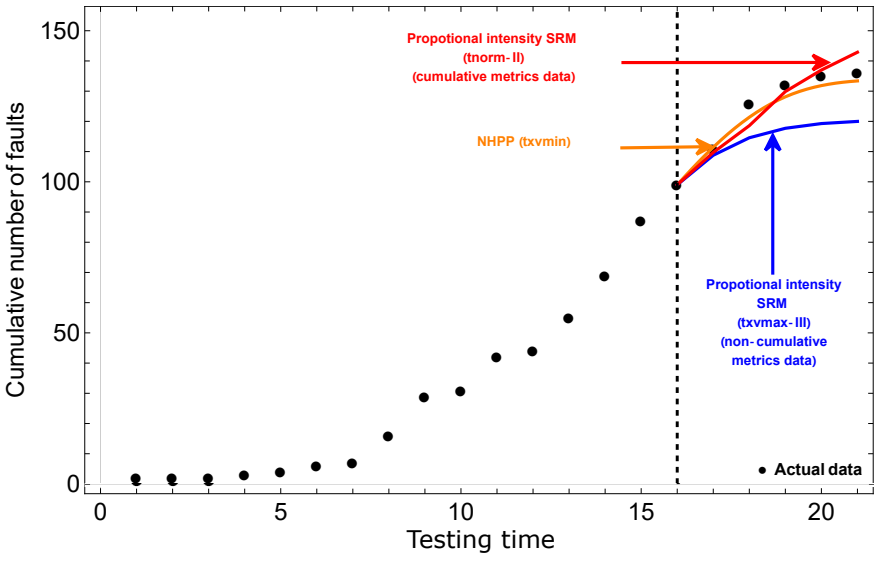


Figure 5.4: Behavior of the predicted cumulative number of software faults with PI-SRMs and common NHPP-based SRM in GDS1 (80% observation point).

Table 5.4: Predictive performance based on PMSE at 50% observation point.

GDS1		
	Best model	PMSE
Case I (cumulative)	tlogist-III	6.409
Case I (non-cumulative)	tlogist-II	4.014
Case II (cumulative)	lxvmax-II	2.160
Case II (non-cumulative)	txvmax-IV	4.931
Case III (cumulative): Linear regression	exp-IV	4.146
Case III (cumulative): Exponential regression	txvmin-V	19.213
Case III (non-cumulative): Linear regression	txvmax-II	3.916
SRATS	tnorm	3.408
GDS2		
	Best model	PMSE
Case I (cumulative)	tlogist-II	0.816
Case I (non-cumulative)	tnorm-III	0.799
Case II (cumulative)	gamma-II	0.742
Case II (non-cumulative)	txvmax-II	0.407
Case III (cumulative): Linear regression	tlogist-IV	0.616
Case III (cumulative): Exponential regression	tnorm-III	1.644
Case III (non-cumulative): Linear regression	tlogist-IV	0.780
SRATS	tlogist	1.769
GDS3		
	Best model	PMSE
Case I (cumulative)	tlogist-II	2.676
Case I (non-cumulative)	txvmax-III	0.481
Case II (cumulative)	exp-VII	0.467
Case II (non-cumulative)	pareto-VI	1.506
Case III (cumulative): Linear regression	llogist-II	0.748
Case III (cumulative): Exponential regression	lxvmax-VI	1.842
Case III (non-cumulative): Linear regression	lxvmax-VII	1.769
SRATS	exp	1.836
GDS4		
	Best model	PMSE
Case I (cumulative)	tlogist-III	2.088
Case I (non-cumulative)	pareto-II	1.506
Case II (cumulative)	exp-I	0.495
Case II (non-cumulative)	tnorm-VI	0.425
Case III (cumulative): Linear regression	txvmax-VI	1.139
Case III (cumulative): Exponential regression	exp-II	0.688
Case III (non-cumulative): Linear regression	lxvmin-I	0.703
SRATS	tlogist	1.754

Table 5.5: Predictive performance based on PMSE at 80% observation point.

GDS1		
	Best model	PMSE
Case I (cumulative)	tnorm-II	2.482
Case I (non-cumulative)	txvmax-III	1.768
Case II (cumulative)	txvmax-VII	2.142
Case II (non-cumulative)	txvmax-V	2.903
Case III (cumulative): Linear regression	tnorm-II	1.033
Case III (cumulative): Exponential regression	tlogist-VII	3.159
Case III (non-cumulative): Linear regression	txvmax-VII	3.916
SRATS	txvmin	1.218
GDS2		
	Best model	PMSE
Case I (cumulative)	pareto-IV	0.488
Case I (non-cumulative)	gamma-V	0.277
Case II (cumulative)	lnorm-VII	0.399
Case II (non-cumulative)	pareto-I	0.466
Case III (cumulative): Linear regression	exp-IV	0.455
Case III (cumulative): Exponential regression	llogist-VI	0.499
Case III (non-cumulative): Linear regression	llogist-IV	0.508
SRATS	lnorm	0.531
GDS3		
	Best model	PMSE
Case I (cumulative)	tnorm-II	0.326
Case I (non-cumulative)	txvmax-II	0.150
Case II (cumulative)	txvmax-IV	0.330
Case II (non-cumulative)	lxvmax-II	0.982
Case III (cumulative): Linear regression	lxvmin-I	0.340
Case III (cumulative): Exponential regression	txvmin-VI	1.484
Case III (non-cumulative): Linear regression	pareto-III	0.293
SRATS	exp	0.295
GDS4		
	Best model	PMSE
Case I (cumulative)	exp-I	0.213
Case I (non-cumulative)	lxvmin-V	0.227
Case II (cumulative)	tnorm-IV	0.220
Case II (non-cumulative)	tnorm-II	0.206
Case III (cumulative): Linear regression	tlogist-II	0.207
Case III (cumulative): Exponential regression	lxvmax-III	0.273
Case III (non-cumulative): Linear regression	tlogist-VII	0.220
SRATS	gamma	0.230

5.3.3 Software Reliability Assessment

In the previous argument, we have confirmed that our PI-SRMs could show better predictive performances than the existing NHPP-SRMs in all cases. In the next step, we wish to quantify the software reliability, which is defined as the probability that the software after release is fault-free. Let $R(t_l \mid t_m) = Pr\{N(t_m) - N(t_l) = 0 \mid N(t_l) = n\}$ denote the software reliability in the operational phase $(t_l, t_m]$, where t_l is the release point. Then, from the NHPP assumption, it is easy to obtain:

$$R(t_l \mid t_m) = \exp\left[M_p(t_m; \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}}) - M_p(t_l; \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}})\right]. \tag{5.15}$$

In our numerical example, we set $t_m = 2t_l$, say, the operational period is twice the length, and assume that the software metrics $\boldsymbol{x}_k = (x_{k1}, x_{k2}, x_{k3})$ are constant in the time interval $(t_l, t_m]$, since the software product has not been tested after the release time t_l. We assess the software reliability quantitatively with the best PI-SRMs, which are selected with the minimum AIC at the release time point $t_l = t_n$.

Table 5.6 presents the comparison results of our PI-SRMs with the existing NHPP-based SRMs. It can be seen that our PI-SRMs with cumulative/non-cumulative software metrics could provide larger software reliability than the common NHPP-based SRMs without software metrics. This result implies that if the PI-SRMs are reliable in goodness-of-fit and predictive performances, they are more inclined to provide positive decisions in terms of software reliability assessment, and the NHPP-based SRMs without software metrics tend to underestimate the software reliability. On the other hand, we also note that in all four data sets, the software reliability estimated by almost all of the SRMs, except in txvmin-II PI-SRM in GDS3 and txvmin NHPP-based SRM in GDS4, are not promising. This observation also implies that in time interval $(t_l, t_m]$, these SRMs tend to give false alarms from the viewpoint of safety, so that the software products under testing seem to require more tests to meet the software reliability requirement.

Table 5.6: Software reliability assessment with best SRM (minimum AIC).

(i) Best proportional intensity model (cumulative metrics data)		
	Model	Reliability
GDS1	tlogist-VI	2.969×10^{-2}
GDS2	tlogist-III	9.260×10^{-1}
GDS3	txvmin-II	9.998×10^{-1}
GDS4	exp-I	5.455×10^{-3}
(ii) Best proportional intensity model (non-cumulative metrics data)		
GDS1	txvmin-II	$4.393 \times \mathbf{10^{-1}}$
GDS2	llogist-II	1.984×10^{-2}
GDS3	gamma-II	2.945×10^{-1}
GDS4	exp-VI	4.324×10^{-1}
(iii) Best SRATS (no metrics data)		
GDS1	tlogist	6.977×10^{-5}
GDS2	llogist	4.152×10^{-3}
GDS3	lxvmax	7.236×10^{-5}
GDS4	txvmin	9.559×10^{-1}

Chapter 6

Non-homogeneous Markov Process-based Software Reliability Models

In this chapter, we focus on non-homogeneous Markov processes (NHMPs), which are generalizations of the well-known HMP- and NHPP-based SRMs, and propose a unified approach to treat the software reliability prediction and its related problems. More specifically, we pay our attention to two subclasses of NHMP; a generalized binomial process (GBP) and a generalized Polya process (GPP), where GBP and GPP can be characterized respectively as a Markov inverse death process [105] and a Markov birth process [106], with state- and time-dependent transition rates. Shanthikumar [107] first considered a GBP-based SRM to unify two well-known SRMs; Goel and Okumoto exponential NHPP-based SRM [10] and an inverse death process-based SRM by Jelinski and Moranda [4]. Since then, NHMP-based SRMs have not been discussed sufficiently in the literature, in spite of its flexibility and applicability in software reliability modeling. The main purpose of this chapter is to provide a unified framework based on both GBP and GPP, to study their features in software reliability modeling, and to compare the goodness-of-fit and predictive performances of those SRMs in a comprehensive empirical study with real software fault count data.

6.1 Preliminary

NHMPs are straightforward extensions of HMPs and NHPPs in the sense that the transition rate depends on both state and time, but possess a more general representation ability to describe the underlying stochastic nature than both of them. In reliability engineering, Smotherman and Zemoude [108] and Smotherman and Geist [109] analyzed component-based phased-mission systems with NHMP and non-homogeneous Markov reward process, respectively. Cosulic et al. [110] evaluated the system reliability of a multi-voltage high-speed train with a simple three-state NHMP. Koutras et al. [111, 112] described intertemporal behaviors of the amount of free physical memory in a software system and of the allocation of server resources in a server client system by NHMPs, respectively. Gokhale et al. [113, 114, 115] considered a software fault detection and removal process with NHMPs and applied a piecewise constant approximation to calculate the expected number of fixed software faults. However, it is worth mentioning that the above references completely lacked a discussion on statistical inference. Since the likelihood-based methods are commonly used to estimate the model parameters from software fault count data, a class of NHMP-based SRMs, which can be used for the data analysis, are rather limited. Our purpose here is to develop estimable NHMP-based SRMs with software fault count data. For numerical computation methods, including the ordinary differential equation method and the uniformization of NHMPs, see Chapter 13 in Trivedi and Bobbio [21].

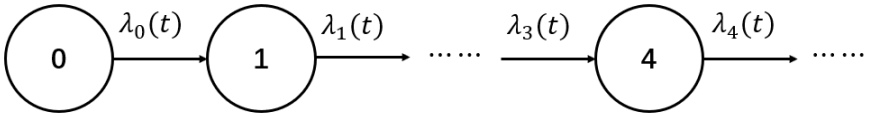
To our best knowledge, Schick and Wolverton [116, 117] and Wagoner [118] treated NHMP-based SRMs for the first time, though they wrongly mentioned that their models were categorized into semi-Markov processes. They extended the seminal Jelinski and Moranda HMP-based SRM [4] by introducing the time-dependent hazard rates, and analyzed only the time-domain data on the software fault count. As another significant contribution following Jelinski and Moranda [4], Littlewood [119, 120] proposed a series of Bayesian SRMs in the so-called hazard rate modeling framework for the time-domain data analysis on the software fault count. Although the statistical inference scheme in [119, 120] is based on the Bayesian approach, the posterior transition rate is time- and state-dependent, and can be viewed as an NHMP. It is shown that the above

SRMs are all classified into the hazard rate type SRMs in some well-referenced text books [2, 3], but belong to a wide class of NHMP-based SRMs.

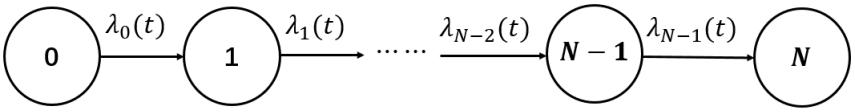
Shanthikumar [107] attempted to unify two well-known SRMs; Jelinski and Moranda HMP-based SRM [4] and Goel and Okumoto exponential NHPP-based SRM [10] by introducing a Markov process under an exponential fault-detection time assumption, and derived the likelihood function with the time-domain data and group data. This unified SRM is often called the Markovian SRM or the binomial SRM in the literature [2, 3], but its mathematical basis was originally given by Kendall [105]. In the subsequent paper [121], it was recognized as a subset of NHMP under an exponential distribution assumption, but we refer to as a generalized binomial process (GBP) in this chapter. We develop new GBP-based SRMs by relaxing the exponential assumption [107, 121]. Another important subclass of NHMP is a generalized Polya process (GPP). Konno [122] argued three Polya processes which denote mixed Poisson processes with gamma mixture [123]. Cha [124] further gave more general mathematical results related to reliability theory in an elegant fashion. We also apply the GPPs to software reliability modeling and develop new GPP-based SRMs as well.

6.2 NHMP-based Software Reliability Modeling

A natural extension of the HMP- and the NHPP-based SRMs is to consider the transition rate depending on both state and time, say, $\lambda_n(t)$ $(n = 0, 1, 2, \cdots)$. We call such a stochastic process the non-homogeneous Markov process (NHMP) in this chapter. Inspired from the existing HMP-based SRMs with and without termination in Figure 1.1 (a) and (b), we consider two types of NHMP-based SRMs. Figure 6.1 (a) and (b) are the transition diagrams of the NHMP with and without an absorving state N. In general, it is difficult to get an analytical solution of the associated Kolmogorov forward equations for an arbitrary $\lambda_n(t)$. Then we need to specify the functional form of the transition rate. The most plausible assumption is the decomposition between the state-dependent term and the time-dependent term, so we suppose the following two linear transition



(a) NHMP with termination.



(b) NHMP without termination.

Figure 6.1: Transition diagrams of NHMPs.

rates:

$$\lambda_n(t) = (\zeta - \epsilon n)\,\kappa(t),\ n = 0, 1, 2, \cdots, \boldsymbol{\gamma}_{\epsilon,\zeta}, \tag{6.1}$$

$$\lambda_n(t) = (\epsilon n + \zeta)\,\kappa(t),\ n = 0, 1, 2, \cdots, \tag{6.2}$$

where ϵ (≥ 0) and ζ (≥ 0) are non-negative real parameters, $\kappa(t)$ is an arbitrary continuous function of t, $\boldsymbol{\gamma}_{\epsilon,\zeta} = \lceil \zeta/\epsilon \rceil$, and $\lceil x \rceil$ is the ceiling function which is the largest integer less than x. In this chapter we call $\kappa(t)$ and $\Lambda(t) = \int_0^t \kappa(x)dx$ *the baseline intensity function* and *the cumulative baseline intensity function*, respectively. The above linear NHMPs have been discussed by Kendall [105] and Konno [122]. Especially, we call the transition rates in Equations (6.1) and (6.2) *the generalized binomial process* (GBP) and *the generalized Polya process* (GPP) respectively.

6.2.1 GBP-based SRMs

Under a specific exponential assumption, Shanthikumar [107, 121] considered an inverse death-process type of NHMP with $\lambda_n(t) = (\zeta - \epsilon n)\,\kappa(t)$ ($n = 0, 1, 2, \cdots, \boldsymbol{\gamma}_{\epsilon,\zeta}$) and $\kappa(t) = \omega b \exp(-bt)$. The Kolmogorov forward equations

in our generalized case are given by

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}P_0(t) &= -\zeta\kappa(t)P_0(t), && (6.3)\\
\frac{\mathrm{d}}{\mathrm{d}t}P_n(t) &= \big\{\zeta-\epsilon(n-1)\big\}\kappa(t)P_{n-1}(t), \\
&\quad -\big\{\zeta-\epsilon n\big\}\kappa(t)P_n(t), \\
&\qquad n=0,1,\cdots,\boldsymbol{\gamma}_{\epsilon,\zeta} && (6.4)\\
\frac{\mathrm{d}}{\mathrm{d}t}P_{\boldsymbol{\gamma}_{\epsilon,\zeta}}(t) &= \big\{\zeta-\epsilon(\boldsymbol{\gamma}_{\epsilon,\zeta}-1)\big\}\kappa(t)P_{\boldsymbol{\gamma}_{\epsilon,\zeta}-1}(t). && (6.5)
\end{aligned}$$

Solving the difference-differential equations in Equations (6.3) $\sim$ (6.5) with the boundary conditions; $P_0(0)=1$ and $P_n(0)=0$ for arbitrary n $(=1,2,\cdots,\boldsymbol{\gamma}_{\epsilon,\zeta})$, yields

$$P_n(t)=\binom{\boldsymbol{\gamma}_{\epsilon,\zeta}}{n}A(t)^{\boldsymbol{\gamma}_{\epsilon,\zeta}-n}(1-A(t))^n, n=0,1,\cdots,\boldsymbol{\gamma}_{\epsilon,\zeta}, \tag{6.6}$$

where

$$\mathrm{A}(t)=e^{-\epsilon\Lambda(t)} \tag{6.7}$$

and

$$\Lambda(t)=\int_0^t\kappa(x)dx. \tag{6.8}$$

It is worth noting that the distribution in Equation (6.6) is also a binomial distribution having

$$\mathrm{E}\left[N(t)\right]=\boldsymbol{\gamma}_{\epsilon,\zeta}\left\{1-e^{-\epsilon\Lambda(t)}\right\}, \tag{6.9}$$

$$\mathrm{Var}\left[N(t)\right]=\boldsymbol{\gamma}_{\epsilon,\zeta}\left\{1-e^{-\epsilon\Lambda(t)}\right\}e^{-\epsilon\Lambda(t)}. \tag{6.10}$$

From Equations (6.9) and (6.10), it always holds that $\mathrm{E}\left[N(t)\right]>\mathrm{Var}\left[N(t)\right]$, so that GBP has *the under-dispersion* property. This is because the variance in Equation (6.10) is decreasing in t and the uncertainty after elapsing the software testing time is reduced. It is obvious that when $\epsilon=0$ and $\kappa(t)=1$ the p.m.f.s in Equation (6.6) are essentially reduced to ones of NHPP-based SRM with intensity function $\kappa(t)$ and HMP-based SRM by Jelinski and Moranda [4], respectively. Table 6.1 presents the *baseline models* considered in the GBP-based SRMs, which are referred to as the mean value function in type-I NHPP in Table

3.1. The last SRM in Table 6.1 was recently introduced in [9] as a generalization of the well-known inflection S-shaped NHPP-based SRM [15]. However, this model is identical to Xiao's SRM [18], where the underlying fault-detection time distribution is given by a Marshall-Olkin type of Weibull distribution [125]. By substituting the baseline model in Table 6.1 with $\Lambda(t)$ in Equation (6.8), we can obtain fourteen GBP-based SRMs corresponding to the existing NHPP-based SRMs.

If the cumulative baseline intensity function $\Lambda(t)$ is bounded, i.e., $\lim_{t\to\infty}\Lambda(t) = \omega$, then $\lim_{t\to\infty}\mathrm{E}[N(t)] = \boldsymbol{\gamma}_{\epsilon,\zeta}\{1-\exp(-\epsilon\omega)\}$ and $\lim_{t\to\infty}\mathrm{Var}[N(t)] = \boldsymbol{\gamma}_{\epsilon,\zeta}\exp(-\epsilon\omega)\{1-\exp(-\epsilon\omega)\}$, otherwise, i.e. if $\lim_{t\to\infty}\Lambda(t)\to\infty$, then $\lim_{t\to\infty}\mathrm{E}[N(t)] = \boldsymbol{\gamma}_{\epsilon,\zeta}$ and $\lim_{t\to\infty}\mathrm{Var}[N(t)] = 0$. Noting that the GBP terminates in $N(t) = \boldsymbol{\gamma}_{\epsilon,\zeta}$ in the sense of sample path, it is seen for the bounded cumulative baseline intensity function $\Lambda(t)$ that $\lim_{t\to\infty}P_{\boldsymbol{\gamma}_{\epsilon,\zeta}}(t) = \{1-\exp(-\epsilon\omega)\}^{\boldsymbol{\gamma}_{\epsilon,\zeta}}$, so there exists a positive probability that all inherent software faults cannot be detected over an infinite testing time. More precisely, it can be found that each transition time distribution from state n to $n+1$ ($n = 0, 1, \ldots, \boldsymbol{\gamma}_{\epsilon,\zeta}-1$) is defective similar to the NHPP-based SRMs with bounded mean value function. In the bounded case, the stochastic process $N(t)$ asymptotically converges to $\boldsymbol{\gamma}_{\epsilon,\zeta}$, say, $\lim_{t\to\infty}N(t) = \boldsymbol{\gamma}_{\epsilon,\zeta}$ with probability one. Hence $\boldsymbol{\gamma}_{\epsilon,\zeta}\{1-\exp(-\epsilon\omega)\}$ is interpreted as the net cumulative number of *detectable* software faults. On the other hand, in the unbounded cumulative baseline intensity function, $N(t)$ converges to $\boldsymbol{\gamma}_{\epsilon,\zeta}$ from the law of large number. The following example, enables us to understand how the unbounded cumulative intensity function was applied in the existing SRMs.

The well-known SRMs by Schick and Wolverton [117] and Wagoner [118] belong to the GBP-based SRMs. In the power-law type baseline model $\Lambda(t) = at^b$ in Table 6.1, we have $\kappa(t) = (a/(1+b))t^{b+1}$ and $\lambda_n(t) = (\zeta-\epsilon n)(a/(1+b))t^{b+1}$. Putting $b' = 1+b$, $\zeta' = (a/b')\zeta$ and $\epsilon' = (a/b')\epsilon$, we get $\lambda_n(t) = (\zeta'-\epsilon' n)t^{b'}$. When $b' = 2$ ($b = 1$) and $b' > 2$ ($b > 1$), the underlying intensity functions coincide Schick and Wolverton SRM [117] and Wagoner SRM [118], respectively. Littlewood [119, 120] considered a pareto type software fault-detection time, and obtained the posterior representation of the transition rate; $\lambda_n(t) = (\zeta-\epsilon n)/(\gamma+t)$ ($n = 0, 1, 2, \cdots, \boldsymbol{\gamma}_{\epsilon,\zeta}$). It is obvious that Littlewood SRM

[119, 120] is also defined as a GBP, where $\kappa(t) = (t+\gamma)^{-1}$ and $\Lambda(t) = \ln(1+t/\gamma)$, which can be viewed as an extension of the logarithmic Poisson execution time SRM [14]. In this case, the expected cumulative number of software faults is given by $\mathrm{E}[N(t)] = \boldsymbol{\gamma}_{\epsilon,\zeta}\{1 - \gamma(\gamma + t)^{\epsilon}\}$.

6.2.2 GPP-based SRMs

Next, we consider the case with $\lambda_n(t) = (\epsilon n + \zeta)\,\kappa(t)$ $(n = 0, 1, 2, \cdots)$. Konno [122] considered a birth-process type of NHMP and formulated the Kolmogorov forward equations:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}P_0(t) &= -\zeta\kappa(t)P_0(t), && (6.11)\\ \frac{\mathrm{d}}{\mathrm{d}t}P_n(t) &= \big\{\epsilon(n-1)+\zeta\big\}\kappa(t)P_{n-1}(t) \\ &\quad -\big\{\epsilon n+\zeta\big\}\kappa(t)P_n(t), \quad n = 1, 2, \cdots . && (6.12) \end{aligned}$$

By solving the difference-differential equations with the initial conditions $P_0(0) = 1$ and $P_n(0) = 0$, it turns out that

$$P_n(t) = \binom{n + \frac{\zeta}{\epsilon} - 1}{n} A(t)^{\frac{\zeta}{\epsilon}}(1 - A(t))^n, \quad n = 0, 1, \cdots . \tag{6.13}$$

Hence, the resulting p.m.f. is a negative binomial distribution with

$$\mathrm{E}\left[N(t)\right] = \boldsymbol{\gamma}_{\epsilon,\zeta}\left\{e^{\epsilon\Lambda(t)} - 1\right\}, \tag{6.14}$$

$$\mathrm{Var}\left[N(t)\right] = \boldsymbol{\gamma}_{\epsilon,\zeta}e^{\epsilon\Lambda(t)}\left\{e^{\epsilon\Lambda(t)} - 1\right\}. \tag{6.15}$$

Konno [122] and Gat [126] called this NHMP *the generalized Polya process* (GPP) and *the generalized Yule process*, respectively. In this chapter, we refer to it as GPP as an alternative of GBP. From Equations (6.14) and (6.15), it is seen that $\mathrm{E}\left[N(t)\right] < \mathrm{Var}\left[N(t)\right]$, so that GBP possesses *the over-dispersion* property which is the plausible feature to describe the uncertainty in the time series analysis.

If $\lim_{t\to\infty}\Lambda(t) = \omega$, then $\lim_{t\to\infty}\mathrm{E}[N(t)] = \boldsymbol{\gamma}_{\epsilon,\zeta}\{\exp(\epsilon\omega) - 1\}$ and $\lim_{t\to\infty}\mathrm{Var}[N(t)] = \boldsymbol{\gamma}_{\epsilon,\zeta}\exp(\epsilon\omega)\{\exp(\epsilon\omega) - 1\}$, otherwise, $\lim_{t\to\infty}\mathrm{E}[N(t)] = \lim_{t\to\infty}\mathrm{Var}[N(t)] \to \infty$. Although the GPP does not have an absorbing state, it is interesting to find in the bounded cumulative baseline intensity case that the expected cumulative number of software faults approaches to $\boldsymbol{\gamma}_{\epsilon,\zeta}\{\exp(\epsilon\omega) - 1\}$, instead of $\lim_{t\to\infty}\Lambda(t) = \omega$ in the NHPP-based SRMs. It should be noted that

GPP is also an extension of a mixed Poisson process model by Okamura and Dohi [123].

It is obvious that GBP and GPP are generalizations of HMP and NHPP so far. Since the p.m.f.s in Equations (6.6) and (6.13) are continuous at $\epsilon = 0$ (see [126]), HMP and NHPP are special cases of GBP and GPP, respectively, when $\kappa(t) = 1$ and $\epsilon = 0$. Figure 6.2 shows the schematic behaviors of intensity functions $\lambda_n(t)$ based on GBP and GPP in a testing period. We can clearly find that whenever software faults have been found, the intensity functions are described as piecewise continuous functions with upward/downward jumps.

6.2.3 Maximum Likelihood Estimation

Here we derive the log likelihood functions of GBP and GPP when the time-domain data or the group data is observed. Let $\boldsymbol{\theta}$ be a free parameter vector involved in the baseline functions $\kappa(t) = \kappa(t;\boldsymbol{\theta})$ and $\Lambda(t) = \Lambda(t;\boldsymbol{\theta})$. Suppose that the fault-detection time-domain data $(t_1, t_2, \cdots, t_m; t_e)$ with censoring point t_e are available. Then, the likelihood function of GBP is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{m} \{\zeta - \epsilon(i-1)\}\kappa(t_i;\boldsymbol{\theta}) \times e^{-\int_0^{t_e}\{\zeta-\epsilon m\}\kappa(x;\boldsymbol{\theta})dx}. \tag{6.16}$$

From a few algebraic manipulations, we have

$$\begin{aligned}
&\int_0^{t_e} \{\zeta - \epsilon m\}\kappa(x;\boldsymbol{\theta})dx \\
&\quad = \sum_{i=1}^{m} \int_{t_{i-1}}^{t_i} \{\zeta - \epsilon(i-1)\}\kappa(x;\boldsymbol{\theta})dx + \int_{t_m}^{t_e} \{\zeta - \epsilon m\}\kappa(x;\boldsymbol{\theta})dx \\
&\quad = \zeta\Lambda(t_e;\boldsymbol{\theta}) - \epsilon\left[m\Lambda(t_e;\boldsymbol{\theta}) - \sum_{i=1}^{m}\Lambda(t_i;\boldsymbol{\theta})\right].
\end{aligned} \tag{6.17}$$

Also, the factorial part in Equation (6.16) becomes

$$\prod_{i=1}^{m}\{\zeta - \epsilon(i-1)\}\kappa(t_i;\boldsymbol{\theta}) = \frac{\Gamma\left(\boldsymbol{\gamma}_{\epsilon,\zeta}+1\right)}{\Gamma\left(\boldsymbol{\gamma}_{\epsilon,\zeta}-m\right)}\epsilon^m \prod_{i=1}^{m}\kappa(t_i;\boldsymbol{\theta}). \tag{6.18}$$

From $\Gamma(k+1) = k\Gamma(k)$ with $k > 0$, we derive the likelihood function and its logarithm:

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{\theta}) \\
&= \left\{\prod_{i=1}^{m} \kappa(t_i;\boldsymbol{\theta})\right\} \frac{\Gamma\left(\gamma_{\epsilon,\zeta}+1\right)}{\Gamma\left(\gamma_{\epsilon,\zeta}-m\right)} \epsilon^m \times e^{-\zeta\Lambda(t_e;\boldsymbol{\theta})+\epsilon\left[m\Lambda(t_e;\boldsymbol{\theta})-\sum_{i=1}^{m}\Lambda(t_i;\boldsymbol{\theta})\right]} \\
&= \left\{\prod_{i=1}^{m} \kappa(t_i;\boldsymbol{\theta}) e^{\epsilon\Lambda(t_i;\boldsymbol{\theta})}\right\} \frac{\Gamma\left(\gamma_{\epsilon,\zeta}+1\right)}{\Gamma\left(\gamma_{\epsilon,\zeta}-m\right)} \epsilon^m e^{(\epsilon m-\zeta)\Lambda(t_e;\boldsymbol{\theta})}, \qquad (6.19)
\end{aligned}$$

and

$$\begin{aligned}
&\ln \mathcal{L}(\boldsymbol{\theta}) \\
&= \left[\sum_{i=1}^{m} \ln \kappa(t_i;\boldsymbol{\theta}) + \sum_{i=1}^{m} \epsilon\Lambda(t_i;\boldsymbol{\theta})\right] + \ln[\left(\gamma_{\epsilon,\zeta}\right)! - \left(\gamma_{\epsilon,\zeta}-m-1\right)!] \\
&+ m\ln(\epsilon) + [(\epsilon m-\zeta)\Lambda(t_e;\boldsymbol{\theta})]\,. \qquad (6.20)
\end{aligned}$$

In the failure truncation case, i.e. $t_e = t_m$, we have

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{m} \{\zeta-\epsilon(i-1)\}\kappa(t_i;\boldsymbol{\theta}) \times e^{-\{\zeta-\epsilon(i-1)\}\{\Lambda(t_i;\epsilon,\boldsymbol{\theta})-\Lambda(t_{i-1};\epsilon,\boldsymbol{\theta})\}}, \qquad (6.21)$$

and

$$\begin{aligned}
\ln \mathcal{L}(\boldsymbol{\theta}) =& \sum_{i=1}^{m} \ln\{\zeta-\epsilon(i-1)\} + \sum_{i=1}^{m} \ln \kappa(t_i;\boldsymbol{\theta}) - \sum_{i=1}^{m}\{\zeta-\epsilon(i-1)\} \\
&\times \Big\{\Lambda(t_i;\boldsymbol{\theta}) - \Lambda(t_{i-1};\boldsymbol{\theta})\Big\}. \qquad (6.22)
\end{aligned}$$

Then, the problem is to maximize the log likelihood function $\ln \mathcal{L}(\epsilon,\zeta,\boldsymbol{\theta})$ with respect to $(\epsilon,\zeta,\boldsymbol{\theta})$.

When the group data (τ_i, n_i) $(i = 0, 1, \cdots, m)$ are available, from Equation (6.6) we obtain

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) =& \prod_{i=1}^{m} \binom{\gamma_{\epsilon,\zeta}-n_{i-1}}{n_i-n_{i-1}} \times [e^{-\epsilon\{\Lambda(\tau_i;\boldsymbol{\theta})-\Lambda(\tau_{i-1};\boldsymbol{\theta})\}}]^{\gamma_{\epsilon,\zeta}-n_i} \\
&\times [1-e^{-\epsilon\{\Lambda(\tau_i;\boldsymbol{\theta})-\Lambda(\tau_{i-1};\boldsymbol{\theta})\}}]^{n_i-n_{i-1}}, \qquad (6.23)
\end{aligned}$$

and

$$\begin{aligned}
\ln \mathcal{L}(\boldsymbol{\theta}) =& \sum_{i=1}^{m} \ln\Big\{\Big(\gamma_{\epsilon,\zeta}-n_{i-1}\Big)! - (n_i-n_{i-1})! - \big(\gamma_{\epsilon,\zeta}-n_i\big)!\Big\} \\
&- \sum_{i=1}^{m}\Big\{\gamma_{\epsilon,\zeta}-n_i\Big\} \times \epsilon\Big\{\Lambda(\tau_i;\boldsymbol{\theta}) - \Lambda(\tau_{i-1};\boldsymbol{\theta})\Big\} \\
&+ \sum_{i=1}^{m}(n_i-n_{i-1}) \times \ln\Big[1-e^{-\epsilon\{\Lambda(\tau_i;\boldsymbol{\theta})-\Lambda(\tau_{i-1};\boldsymbol{\theta})\}}\Big]. \qquad (6.24)
\end{aligned}$$

Next, we consider the likelihood functions for GPP. Given the fault-detection time-domain data with the censoring point t_e, the likelihood function and its logarithm are, in a fashion similar to Equations (6.19) and (6.20), given by

$$\mathcal{L}(\boldsymbol{\theta}) = \left\{ \prod_{i=1}^{m} \kappa(t_i;\boldsymbol{\theta}) e^{\epsilon\Lambda(t_i;\boldsymbol{\theta})} \right\} \frac{\Gamma\left(\boldsymbol{\gamma}_{\epsilon,\zeta}+1\right)}{\Gamma\left(\boldsymbol{\gamma}_{\epsilon,\zeta}-m\right)} \frac{\epsilon^m}{e^{(\epsilon m+\zeta)\Lambda(t_e;\boldsymbol{\theta})}}, \tag{6.25}$$

$$\begin{aligned} \ln \mathcal{L}(\boldsymbol{\theta}) = &\left[\sum_{i=1}^{m} \ln \kappa(t_i;\boldsymbol{\theta}) + \sum_{i=1}^{m} \epsilon\Lambda(t_i;\boldsymbol{\theta}) \right] + \ln[\left(\boldsymbol{\gamma}_{\epsilon,\zeta}\right)! \\ &- \left(\boldsymbol{\gamma}_{\epsilon,\zeta}-m-1\right)!] + (m\ln(\epsilon) - [(\epsilon m+\zeta)\Lambda(t_e;\boldsymbol{\theta})]) . \end{aligned} \tag{6.26}$$

The above results are due to Asfaw and Lindqvist [127]. In the failure truncation case, i.e. $t_e = t_m$, we have

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{m} \{\epsilon(i-1)+\zeta\}\kappa(t;\boldsymbol{\theta}) \times e^{-\{\epsilon(i-1)+\zeta\}\{\Lambda(t_i;\epsilon,\boldsymbol{\theta})-\Lambda(t_{i-1};\epsilon,\boldsymbol{\theta})\}}, \tag{6.27}$$

and

$$\begin{aligned} \ln \mathcal{L}(\boldsymbol{\theta}) = &\sum_{i=1}^{m} \ln\{\epsilon(i-1)+\zeta\} + \sum_{i=1}^{m} \ln \kappa(t_i;\boldsymbol{\theta}) \\ &- \sum_{i=1}^{m} \{\epsilon(i-1)+\zeta\}\Big\{\Lambda(t_i;\boldsymbol{\theta}) - \Lambda(t_{i-1};\boldsymbol{\theta})\Big\}. \end{aligned} \tag{6.28}$$

On the other hand, when the group data (τ_i, n_i) $(i = 0, 1, \cdots, m)$ with GPP are available, then, from Equation (6.13), we get

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) = &\prod_{i=1}^{m} \binom{n_i + \boldsymbol{\gamma}_{\epsilon,\zeta} - 1}{n_i - n_{i-1}} \times [e^{-\epsilon\{\Lambda(\tau_i\boldsymbol{\theta})-\Lambda(\tau_{i-1};\boldsymbol{\theta})\}}]^{\frac{\zeta}{\epsilon}+n_{i-1}} \\ &\times [1 - e^{-\epsilon\{\Lambda(\tau_i;\boldsymbol{\theta})-\Lambda(\tau_{i-1};\boldsymbol{\theta})\}}]^{n_i-n_{i-1}}, \end{aligned} \tag{6.29}$$

and

$$\begin{aligned} \ln \mathcal{L}(\boldsymbol{\theta}) = &\sum_{i=1}^{m} \ln\Big\{\Big(n_i + \boldsymbol{\gamma}_{\epsilon,\zeta} - 1\Big)! - (n_i - n_{i-1})! - \big(n_{i-1} + \boldsymbol{\gamma}_{\epsilon,\zeta} - 1\big)!\Big\} \\ &- \sum_{i=1}^{m} \Big\{\boldsymbol{\gamma}_{\epsilon,\zeta} + n_{i-1}\Big\} \times \epsilon\Big\{\Lambda(\tau_i;\boldsymbol{\theta}) - \Lambda(\tau_{i-1};\boldsymbol{\theta})\Big\} + \sum_{i=1}^{m} (n_i - n_{i-1}) \\ &\times \ln\Big[1 - e^{-\epsilon\{\Lambda(\tau_i;\boldsymbol{\theta})-\Lambda(\tau_{i-1};\boldsymbol{\theta})\}}\Big]. \end{aligned} \tag{6.30}$$

6.3 Performance Comparisons

6.3.1 Data Sets

In numerical experiments, we analyze eight data sets of software fault count time-domain data (see Table 1.1 (i) DS1, DS4, DS8 $\sim$ DS13) from CSS development projects, eight data sets of group data (see Table 1.1 (ii) DS14 *sim* DS21) from CSS development projects and four data sets (see Table 1.1 (iii) DS22 $\sim$ DS25) from OSS development projects. We re-name these data from DS1 to DS20. Especially, it is worth noting that in the OSS development projects, software fault counts are made in accordance with the bug reports provided by the software users. Based on the fourteen models in Table 6.1, we derive the maximum likelihood estimates of the model parameters and obtain the maximum likelihood function $\ln \mathcal{L}(\hat{\epsilon}, \hat{\zeta}, \hat{\boldsymbol{\theta}})$ in NHPP-, GBP- and GPP-based SRMs.

6.3.2 Goodness-of-fit Performances

As a goodness-of-fit measure, we apply the well-known AIC, where $\ln \mathcal{L}(\hat{\epsilon}, \hat{\zeta}, \hat{\boldsymbol{\theta}})$ is the maximized log likelihood function. The smaller AIC is the better SRM in terms of the goodness-of-fit to the underlying fault count data. Note that the difference on the number of free parameters between NHPP and GBP/GPP is at most 1 for each baseline model in Table 6.1. Hence, if the difference of AIC is greater than 2, we can recognize that GBP/GPP is significantly different from NHPP.

In Figure 6.3, we plot the temporal behavior of the cumulative number of software faults with CSS development project (time-domain data) in DS1, CSS development projects (group data) in DS9, and OSS development projects (group data) in DS17, respectively. Looking at the curves (fitted mean value functions), almost all SRMs could catch up the average trend of the cumulative number of software faults detected in the testing phase, but a few SRMs failed to describe realization of the underlying stochastic counting process.

More specifically, in Table 6.2, we compare three modeling frameworks; NHPP, GBP and GPP, in terms of the AICs, where the bold font denotes the best SRM among three modeling frameworks. For the CSS development projects (time-domain data), it is seen that the common NHPP-based SRMs showed better results than both GBP- and GPP-based SRMs with $\epsilon > 0$ in

five data sets but the difference was significant in only DS7. In the other data sets, GBP-based SRM outperformed the NHPP-based SRM and gave the significantly better goodness-of-fit in DS8. Since GBP and GPP reduce to the corresponding NHPP when $\epsilon = 0$, we can conclude that NHPP-, GBP- and GPP-based SRMs show almost similar goodness-of-fit performances in many cases when the CSS projects (time-domain data) are available. For the group data, GBP-based SRM provided the significantly better result in DS11 with $\epsilon > 0$, but the difference on AIC between NHPP-based SRM and GBP/GPP-based SRM was not so remarkable in the remaining data sets. On the other hand, in half of the OSS development projects, GBP performed better than the remaining two frameworks, but only showed a significant difference in DS17.

Next we focus on the best baseline models in Table 6.2. In five out of eight cases in the CSS development projects (time-domain data), four out of eight cases in the CSS development projects (CSS project (group data)) and two out of four cases in the OSS development projects (CSS project (group data)), the best baseline models with minimum AIC are exactly same for three different modeling frameworks. This suggests that the extreme type distributions such as Lxvmax and Txvmin tend to be the best baseline model, and that the classical SRMs like Exp, Llogist and Log are still valid even though GBP/GPP-based SRMs are considered. Meanwhile, we believe that Gtlogist should also be a good selection for the baseline model, as some SRMs based on this probability distribution have shown the superiority of fit for the OSS development projects. Eventually, we find that eight baseline models are enough among fourteen baseline model candidates in Table 6.2, if we are interested in the goodness-of-fit performance with the full data.

6.3.3 Predictive Performances

Next, we investigate the predictive performance of our GBP/GPP-based SRMs with different baseline models. To compare the predictive performance quantitatively, we use the PMSE and set the observation point l at 20%, 50% and 80% points of the whole time series data. Figures 6.4, 6.5, and 6.6 are the predicted results of the cumulative number of software faults in DS1, DS9, and DS17, respectively. As easily expected, the long-term prediction from 20% observation

point is not accurate enough because the trend change may occur in the remaining testing period. Table 6.3 presents the prediction results at each observation point based on the minimum PMSE, where the best SRM with the minimum PMSE in each data set is represented with bold font. For the CSS development projects (time-domain data), when the testing phase is early (20%) and middle (50%), GBP/GPP-based SRMs provided the minimum PMSEs in six cases. In the late stage of testing (80%), GPP-based SRM could show the better predictive performances in seven cases than the NHPP-based one, although five data sets were equivalent in prediction performance. Comparing Table 6.3 with Table 6.2, it is seen that the best baseline models on the goodness-of-fit criterion are rather different from ones minimizing PMSE at each observation point. The baseline model; Lxvmax at 20% observation in DS4 and DS6, and Lxvmax at 80% observation in DS8, was selected as the best for three different modeling frameworks.

In Table 6.4, we summarize the minimum PMSE with the CSS development projects (group data), and compare three modeling frameworks for fourteen baseline models. It is pointed out that the number of cases where NHPP-based SRM strictly outperformed GBP-, GPP-based SRMs was only three (DS10 at 20%, DS14 at 50% and 80%). In the CSS projects (time-domain data), GPP-based SRM tended to give the better predictive performance in many cases (five cases at 20%, four cases at 50%, six cases at 80%). From these results, we recognize that the GPP-based SRMs could show the best prediction performances for both CSS projects (time-domain data) and CSS projects (group data). It is worth mentioning that the GPP-based SRMs have not been fully discussed in the literature. The lesson learned from our numerical experiments suggests that there are no remarkable differences on goodness-of-fit performance in three modeling frameworks, but our GPP-based SRMs have the potential ability to make the accurate prediction for the unknown future pattern on fault detection. On the best baseline models, we found that Plaw in DS15 at 20% and Lxvmax in DS15 and DS16 at 50% are identical for three modeling frameworks.

Focusing on the OSS development projects, we observe the minimum PMSE for each modeling framework in Table 6.5 and provide a comparison of the three modeling frameworks. It can be noticed that GPP-based SRM could guaran-

tee the better predictive performance in three out of the four data sets during the early operational stage (20%) after the OSS was released. However, when the operational phase is gradually extended, GBP- and GPP-based SRMs have hardly outperformed NHPP-based SRMs in terms of predictive performance. They only gave one set of better predicted performance at 50% and 80% observation point, respectively (DS18 at 50% and DS17 at 80%). Similar to the case of the best baseline model shown in Table 6.2, Table 6.5 pointed out that the three modeling frameworks do not often guarantee the same best baseline model in terms of predictive performance for OSS data. Some of the classic baseline models; Log (DS19 at 20% and DS17 at 50%), Plaw (DS20 at 20%, 50% and 80%), Pareto (DS17 at 80%) and Gamma (DS19 at 80%) are still superior enough, even though we had a total of fourteen baseline models.

In Tables 6.3, 6.5 and 6.5, we compared the best SRM with the minimum PMSE in respective data sets. However, It should be noted that one cannot know the best baseline model in advance before making the prediction. The commonly applied method is to make the prediction of the future behavior of software faults by using the best fitted SRM to the past observation, if it was not overfitted. Hence, it is plausible to compare the predictive performances with three modeling frameworks using SRMs having the minimum AIC. Tables 6.6∼6.12 present the predictive performances when the baseline model was selected with the minimum AIC at each observation point, in the CSS projects (time-domain data), CSS projects (group data) and OSS projects (group data), respectively. For the CSS development projects, it could be confirmed that NHPP-based SRMs gave the minimum AIC values in almost all cases except in DS7 at 50%, DS3, DS6 and DS8 at 80% in the time-domain data, DS12 at 20%, DS10, DS11 and DS16 at 50%, DS9, DS11, DS12, DS16 at 80% in group data. However, for the OSS development projects (group data), NHPP-based SRMs only guaranteed the minimum AIC in half or less of the cases, at either observation point. On the other hand, the number of cases where NHPP-based SRM gave both minimum AIC and PMSE was eleven out of twenty four (DS4, DS6 and DS8 at 20%, DS1, DS2, DS7 at 50%, DS1, DS2, DS4, DS5, DS6, DS7 at 80%) in the CSS projects (time-domain data), three out of fifteen (DS10 at 20%, DS9 at 50%, DS10 at 80%) in the CSS projects (group data) and two out

of five (DS18, DS19 at 80%) in the OSS projects (group data). Hence, it is obvious that selecting the best SRM in terms of AIC does not lead to the best SRM with highest prediction ability.

It is worth pointing out in Tables 6.6∼6.12 that the best baseline models in the respective modeling frameworks based on NHPP, GBP or GPP were rather similar. That is to say, if we select the best baseline model among NHPP-based SRMs and apply the same baseline model for prediction with NHMP-based SRMs, GBP and GPP provided the best predictive performance seven cases and nine cases, respectively in the CSS projects (time-domain data). In the CSS projects (group data), GBP and GPP resulted the best predictive performances in seven cases and eleven cases, respectively. In the OSS projects (group data), NHMP could achieve the best prediction performance in eight cases. Though there is no significant difference between NHPP- and GBP/GPP-based SRMs, when NHMP-based SRMs were applied to the same baseline model in the best NHPP-based SRM for the prediction, GBP- and GPP-based SRMs provided the minimum PMSE (including the equivalent cases to the NHPP-based SRMs) in seven and nine cases in the CSS projects (time-domain data), in seven and eleven cases in CSS projects (group data), in two and seven cases in OSS projects (group data). So, it is recommended to select the best baseline model by NHPP-based SRM and to predict the number of software faults detected/reported in the future by GBP- or GPP-based SRM with the same baseline model. At the first look, GPP-based SRM seems to have advantages in terms of the number of better cases, especially in the long-term prediction of OSS projects (group data) (see PMSEs at 20% in Table 6.12), but, for CSS projects, may lead to extremely worse results (see DS2 at 20% in Table 6.6, DS13 and DS16 at 20% in Table 6.9. It is obvious that in the mid-term of software testing, GBP-based SRMs are the better alternative. For both CSS and OSS projects, GBP-based SRMs guaranteed the better predictive performance in at least half of the cases. Finally, we can conclude that our generalized modeling frameworks based on GBP and GPP are superior to the common NHPP-based SRMs in terms of the predictive performance even in actual situations without knowing the real best baseline model at each observation point.

6.3.4 Software Reliability Assessment

Next, we concern to quantify the software reliability. Since the quantitative software reliability, $R_t(x)$, is defined as the probability that software is fault-free in the time interval $(t, t+x]$, provided that the software in the CSS development project is released at time t (≥ 0). On the other hand, in the OSS development project, the time t (≥ 0) is regarded as the operational time length of the software since its release. In our NHMP-based modeling framework, we define the software reliability as

$$R_t(x) = \Pr(N(t+x) - N(t) = 0 \mid N(t) = m), \tag{6.31}$$

where m is the cumulative number of software faults detected/reported by time t. For an infinitesimal time $\triangle x$, we have

$$\begin{aligned}
&R_t(x + \triangle x) \\
&= \Pr(N(t + x + \triangle x) - N(t) = 0 \mid N(t) = m) \\
&= \Pr(N(t + x + \triangle x) - N(t + x) = 0 \mid \\
&\quad N(t + x) - N(t) = 0, N(t) = m) \\
&\quad \times \Pr(N(t + x) - N(t) = 0 \mid N(t) = m) \\
&= \Pr(N(t + x + \triangle x) - N(t + x) = 0 \mid N(t + x) = m) \\
&\quad \times \Pr(N(t + x) - N(t) = 0 \mid N(t) = m),
\end{aligned} \tag{6.32}$$

which is due to the Markov property. From Equations (6.1) and (6.2), we get

$$\begin{aligned}
&R_t(x + \triangle x) - R(x) \\
&= (\zeta - \epsilon m)\kappa(t + x)\triangle x R(x) + o(\triangle x)
\end{aligned} \tag{6.33}$$

and

$$\begin{aligned}
&R_t(x + \triangle x) - R(x) \\
&= (\epsilon m + \zeta)\kappa(t + x)\triangle x R(x) + o(\triangle x),
\end{aligned} \tag{6.34}$$

for GBP- and GPP-based SRMs, respectively, where $\lim_{\triangle x \to 0} o(\triangle x)/\triangle x = 0$. Then, the differential equations which satisfy the quantitative software reliability functions for GBP- and GPP-based SRMs are given by

$$\frac{\mathrm{d}}{\mathrm{d}x} R(x) = -(\zeta - \epsilon m)\kappa(t + x)R(x) \tag{6.35}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}x}R(x) = -(\epsilon m + \zeta)\kappa(t+x)R_t(x), \tag{6.36}$$

respectively. By solving Equations (6.35) and (6.36) with $R_t(0) = 0$, we obtain

$$R(x) = \exp\left\{-(\zeta - \epsilon m)\left[\Lambda(t+x) - \Lambda(t)\right]\right\}, \tag{6.37}$$

$$R(x) = \exp\left\{-(\epsilon m + \zeta)\left[\Lambda(t+x) - \Lambda(t)\right]\right\} \tag{6.38}$$

for GBP- and GPP-based SRMs.

Table 6.13 presents the quantitative software reliability with the CSS development projects (time-domain data), CSS development projects (group data) and OSS development projects (group data), respectively, where we select the best baseline model in terms of the minimum AIC at the observation time points; t_m and τ_m. In these examples, for CSS development projects (time-domain data), we set the operational time x as ten times of the testing time length t_m in CPU time. For CSS development projects (group data) and OSS development projects (group data), we set the operational time x as one year (fifty three weeks or twelve months). In Table 6.13, we denote the largest software reliability value in three modeling frameworks with bold font. It can be seen that the resulting software reliability values were too small. This is because, for CSS development projects (time-domain data) and CSS development projects (group data), we implicitly assume that the software was released to the market just after the observation point, where almost all software faults were detected by the release points in the underlying development projects. On the other hand, for OSS project, the software in the OSS development project is release, and new source code is available through versioning up.

In Figure 6.7 (a), we illustrate the software reliability as a function of the operational time x when DS13 is assumed. It can be observed that the software reliability value drops down to the zero level immediately, even for a relatively short operational time interval. It is clear that such a reliability measure will not be useful to make the release decision. In comparison of three modeling frameworks, it is remarkable that GPP-based SRMs tended to make optimistic estimation of software reliability in most of the cases. If no fault is detected after the observation time points; t_m and τ_m, the smaller reliability is more realistic.

In this sense, the software reliability assessment based on Equations (6.37) and (6.38) seem to be quite problematic. Then, we consider the *virtual testing time* to make more realistic software reliability assessment. At each observation point on software testing progress, we see the virtual testing time in the future. Define the hypothesis: no software fault is experienced during the operational time period. If this subjective hypothesis was true after continuing software testing during the (virtual) software testing time period, then we will release the software at the end of the virtual testing time, and the software operational phase starts from the release point. Otherwise, i.e., if any fault was found during the virtual testing period, then we reset the starting point of the virtual testing time (observation point) after correcting the detected faults. On the other hand, we did not consider this problem for the OSS development projects because there does not exist the testing time for operational phase of the OSS. Let s_v denote the virtual testing time length. Then, the total testing time one expects is given by $t = t_m + s_v$ and $t = \tau_m + s_v$ for the CSS development projects (time-domain data) and CSS development projects (group data), respectively, where no software fault will not be found in $(t_m, t_m + s_v]$ and $(\tau_m, \tau_m + s_v]$, under the hypothesis. In other words, we suppose no fault count from t_m to $t_e = t_m + s_v$ for the CSS development projects (time-domain data) and from τ_m to $\tau_m + s_v$ for the CSS development projects (group data), where $(\tau_{m+j}, n_{m+j}) = (\tau_{m+j}, n_m)$ $(j = 0, 1, 2, \ldots)$ during $(\tau_m, \tau_m + s_v]$. Note that the maximum likelihood estimation is repeatedly made by substituting the truncation time t_e or the zero fault count data $(\tau_{m+j}, n_{m+j}) = (\tau_{m+j}, n_m)$ $(j = 0, 1, 2, \ldots)$ in the log likelihood functions.

In Figure 6.7 (a), (b), (c), (d), we compare the predictive software reliability when the virtual testing time lengths are 0%, 50%, 100% and 150% of the whole testing period in DS13. Comparing with Figure 6.7 (a), the software reliability functions in (b), (c) and (d) gradually decrease. It is evident that the higher software reliability is caused by the fact that the virtual testing time length without fault detection is long enough. This observation gives us a reality such that the experience of tester leads to the reliable software program. When the virtual testing time length is given by a half length of the previous testing time period in (b), the predictive software reliability functions in three modeling frameworks show the similar behavior. However, when the virtual testing time

length was exactly same and 150% of the previous testing time period in (c) and (d), it is found that the GPP-based SRM tends to give a more optimistic prediction of quantitative software reliability. We concentrate only the CSS projects (group data), hereafter. Table 6.14 presents the software reliability prediction under the hypothesis by three different SRMs, when the virtual test time is given by 50%, 100% and 150% of the previous testing time length, where the group data sets DS9 $\sim$ DS16 were analyzed. Compared to Table 6.13, the estimated software reliability value is close to unity under the hypothesis that no software fault was found during the virtual testing time. We also recognize here that the higher software reliability is caused by the experience of zero fault count for the system testing period. Our experiments suggest that the user/customer test should be performed during more than the system testing time on the development side, if the software reliability requirement level is high enough.

6.4 Software Release Decision

In this section, we consider the software release problem. As we have introduced the virtual testing time in previous section, the software release timing can be determined by controlling the virtual testing time so as to satisfy a given reliability requirement level. Following [128, 129, 130, 131, 132], we consider the so-called *software cost model* to determine the software release timing with three kinds of modeling framework; NHPP-, GBP- and GBP-based SRMs. Similar to the software reliability assessment with virtual testing time, the software release decision for the OSS is related with the vision up, but not to terminate the software testing. Hence, we consider only the CSS development projects for the software release problems. Since the seminal work by Okumoto and Goel [130], many software cost models have been discussed in the literature (see *e.g.*, [131] and [132]). Let c_1, c_2 and c_3 denote the system testing cost per unit of time, debugging cost to remove a software fault in the testing phase, and debugging cost to remove a software fault in the operational phase, respectively, where without any loss of generality, $c_3 > c_2$. Suppose that the software system test starts at time $t = 0$ and terminates at time $t = t_0$, so that t_0 is regarded as the software release timing to the user or market. Let $M(t) = \mathrm{E}[N(t)]$ be the

expected cumulative number of software faults detected by time t. Then, the expected total software cost, $C(t_0)$, is formulated as

$$C\left(t_0\right) = c_1 t_0 + c_2 M(t_0) + c_3 \left[M\left(T_{LC} + t_0\right) - M\left(t_0\right)\right], \tag{6.39}$$

where T_{LC} is the software life cycle. Then the problem is to derive the optimal software release time, which minimizes $C(t_0)$:

$$\min_{0 \leq t_0 \leq \bar{t}_0} C(t_0), \tag{6.40}$$

where $\bar{t}_0$ is an upper limit of t_0, which may be interpreted as an arbitrary release bound, say $0 < t_0 \leq \bar{t}_0 < T_{LC}$.

The underlying optimization problem in Equation (6.40) is well-defined, because it takes account of the trade-off relationship between the testing cost and the debugging cost. If the release time t_0 is much shorter, then we may encounter the situation where a number of failures caused by faults occur in the operational phase. On one hand, if the release time t_0 is much longer, the testing cost proportional to the testing time length and the debugging cost in the testing phase increase.

For a cumulative intensity function $\Lambda(t)$ in NHMP-based SRMs, the expected cumulative numbers of software faults detected by time t are given by $M(t) = \gamma_{\epsilon,\zeta}\{1 - \exp(-\epsilon\Lambda(t))\}$ and $M(t) = \gamma_{\epsilon,\zeta}\{\exp(\epsilon\Lambda(t)) - 1\}$, respectively, for GBP- and GPP-based SRMs, from Equations (6.9) and (6.14). Since $\Lambda(t)$ is monotonically increasing in t, $M(t)$ is also increasing in both cases. Further if $\lim_{t\to\infty} \Lambda(t) \to \infty$, then $M(t) \to \gamma_{\epsilon,\zeta}$ in GBP and $M(t) \to \infty$ in GPP, otherwise, $M(t)$ is bounded in both cases. From the above insights, the expected total software cost, $C(t_0)$, has different properties in respective modeling frameworks.

Here, we analyze an actual CSS development project (group data) DS15, where the total number of software faults was 58 and the system testing length was 33 weeks. We estimate the model parameters for 3×14 SRMs by means of the maximum likelihood estimation, and derive the maximum likelihood estimates of the expected cumulative number of software faults $M(t)$. In our numerical experiment, the parameter of each cost component and software life cycle are cited from Okumoto and Goal [130]; $c_1 = 10$ (USD per week), $c_2 = 1$ (USD per fault), $c_3 = 5$ (USD per fault), and $T_{LC} = 100$ (weeks). For each SRM, we select the best baseline function with the minimum AIC, and calculate

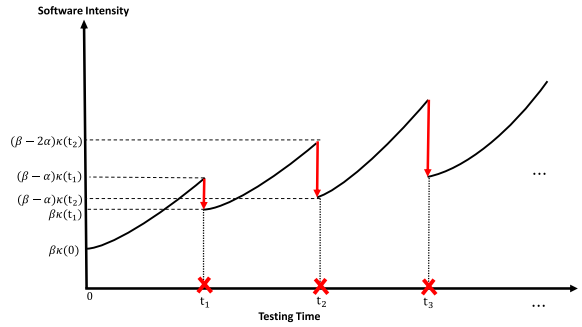
the optimal software release time and its associated expected total software cost numerically.

Table 6.15 presents the comparison results of the optimal software release policies and their associated expected total software costs. We still use bold font to show the best modeling framework (NHPP, GBP or GPP) in the optimal software release time which minimizes most the expected total software cost. The results show that the minimum expected total software cost strongly depends on the kind of framework. Looking at the difference of baseline models, the minimum expected total software costs were rather different from each other. In our example, we assume to predict the expected total software cost at 33rd week and want to know when stopping the software testing. In this sense, the feasible SRMs were just 10 models; NHPP (Exp), NHPP (Log), GBP (Pareto), GBP (Lnorm), GBP (Lxvmax), GBP (Log), GPP (Exp), GPP (Pareto), GPP (Lxvmax), because the corresponding optimal software release times were less than 33 weeks. From Table 6.2, we know that the baseline model Txvmax was the best for three modeling frameworks and the goodness-of-fit performances were almost similar. In this situation, since the optimal software release times were all less than 33 weeks in Table 6.15, it is easily checked that the expected total software cost increases in t_0. To this end, the optimal decision is to stop the software testing at the 33-rd week in our example.

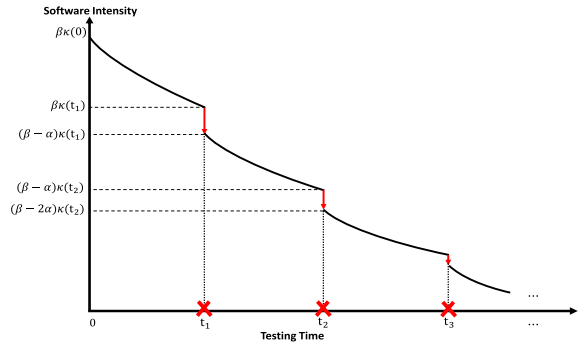
Table 6.1: Representative baseline models in SRMs.

Models	$\Lambda(t)$
Exponential dist. (Exp) [10]	$\Lambda(t) = \omega F(t),\ F(t) = 1 - e^{-bt}$
Gamma dist. (Gamma) [19],[20]	$\Lambda(t) = \omega F(t),\ F(t) = \int_0^t \frac{c^b s^{b-1} e^{-cs}}{\Gamma(b)} ds$
Pareto dist. (Pareto) [7]	$\Lambda(t) = \omega F(t),\ F(t) = 1 - (\frac{b}{t+b})^c$
Truncated normal dist. (Tnorm) [17]	$\Lambda(t) = \omega \frac{F(t)-F(0)}{1-F(0)},\ F(t) = \frac{1}{\sqrt{2\pi}b} \int_{-\infty}^t e^{-\frac{(s-c)^2}{2b^2}} ds$
Log-normal dist. (Lnorm) [8],[17]	$\Lambda(t) = \omega F(\ln t),\ F(t) = \frac{1}{\sqrt{2\pi}b} \int_{-\infty}^t e^{-\frac{(s-c)^2}{2b^2}} ds$
Truncated logistic dist. (Tlogist) [15]	$\Lambda(t) = \omega F(t),\ F(t) = \frac{1-e^{-bt}}{1+ce^{-bt}}$
Log-logistic dist. (Llogist) [13]	$\Lambda(t) = \omega F(\ln t),\ F(t) = \frac{(bt)^c}{1+(bt)^c}$
Truncated extreme-value max dist. (Txvmax) [16]	$\Lambda(t) = \omega \frac{F(t)-F(0)}{1-F(0)},\ F(t) = e^{-e^{-\frac{t-c}{b}}}$
Log-extreme-value max dist. (Lxvmax) [16]	$\Lambda(t) = \omega F(\ln t),\ F(t) = e^{-(\frac{t}{b})^{-c}}$
Truncated extreme-value min dist. (Txvmin) [16]	$\Lambda(t) = \omega \frac{F(0)-F(t)}{F(0)},\ F(t) = e^{-e^{-\frac{t-c}{b}}}$
Log-extreme-value min dist. (Lxvmin) [12]	$\Lambda(t) = \omega\,(1 - F(-\ln t)),\ F(t) = e^{-e^{-\frac{t-c}{b}}}$
Logarithmic Poisson (Log) [2],[14]	$\Lambda(t) = a \ln(1 + bt)$
Power-law (Plaw) [11],[47],[48]	$\Lambda(t) = at^b$
Generalized truncated logistic dist. (Gtlogist) [9],[18]	$\Lambda(t) = \omega \frac{1 - e^{-at^c}}{1 + be^{-at^c}},\ F(t) = \frac{1 - e^{-at^c}}{1 + be^{-at^c}}$

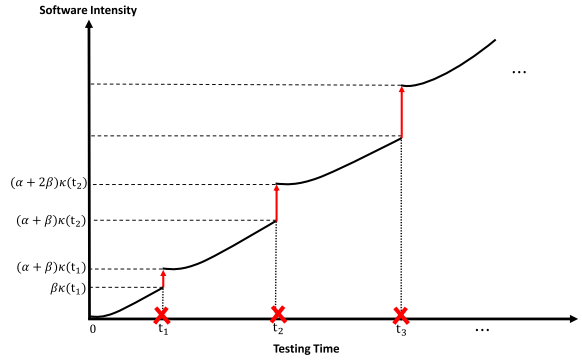
(ω>0, a>0, b>0, c>0)



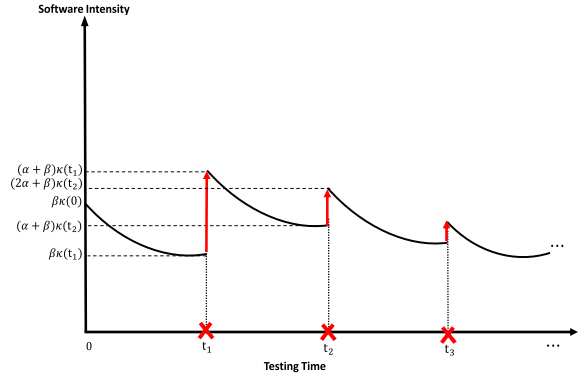
(a) Piecewise increasing intensity function of GBP-based model with downward jump.



(b) Piecewise decreasing intensity function of GBP-based model with downward jump.



(c) Piecewise increasing intensity function of GPP-based model with upward jump.

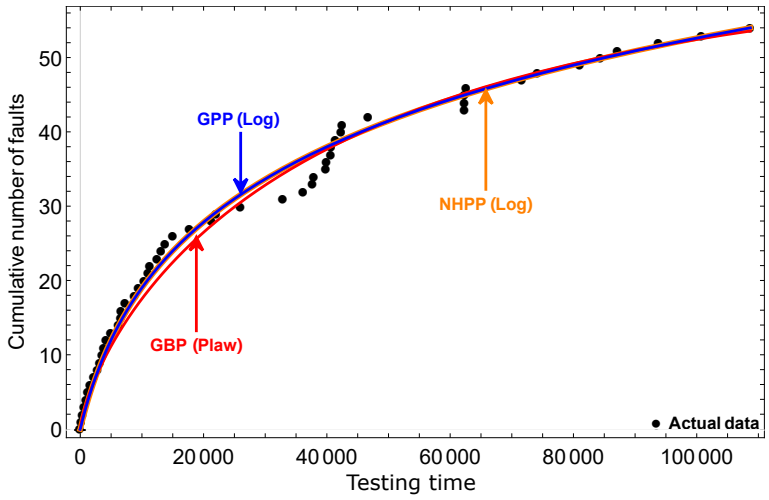


(d) Piecewise decreasing intensity function of GPP-based model with upward jump.

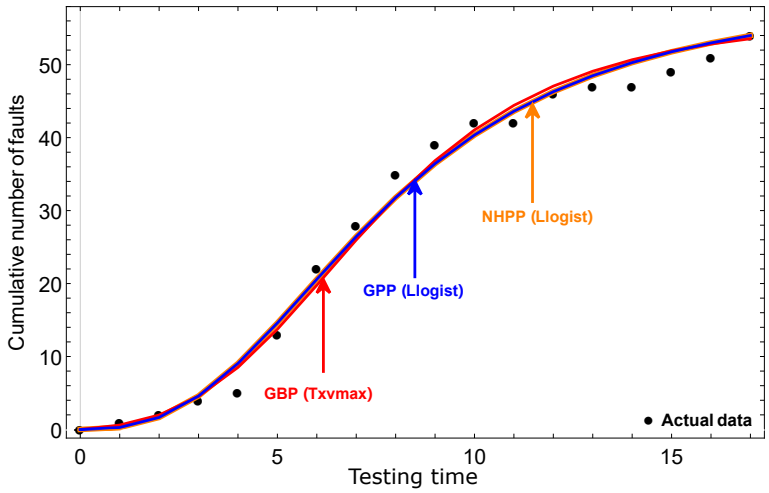
Figure 6.2: Behavior of intensity function $\lambda_n(t)$ in the NHMP-based SRMs.

Table 6.2: Goodness-of-fit performances based on AIC.

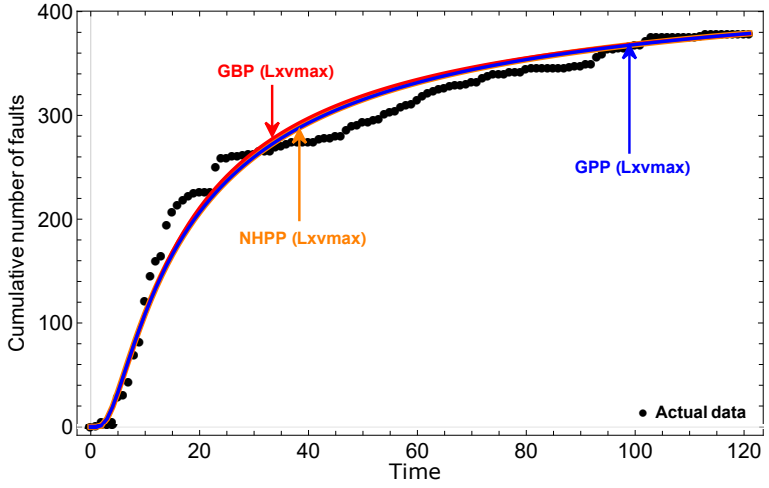
CSS development projects (time-domain data)						
	NHPP		GBP		GPP	
	Best Model	AIC	Best Model	AIC	Best Model	AIC
DS1	Log	**895.305**	Plaw	896.668	Log	897.305
DS2	Plaw	**234.967**	Tlogist	235.343	Plaw	236.967
DS3	Log	**1007.100**	Log	1007.723	Log	1009.100
DS4	Lxvmax	**1822.000**	Lxvmax	1823.876	Lxvmax	1824.000
DS5	Txvmin	5296.620	Txvmin	**5296.000**	Txvmin	5298.620
DS6	Lxvmin	1938.160	Plaw	**1936.541**	Lxvmin	1940.160
DS7	Log	**1203.523**	Log	1205.529	Log	1205.093
DS8	Lxvmax	4720.070	Lxvmax	**4715.180**	Lxvmax	4722.070
CSS development projects (group data)						
	NHPP		GBP		GPP	
	Best Model	AIC	Best Model	AIC	Best Model	AIC
DS9	Llogist	**73.053**	Txvmax	73.187	Llogist	75.053
DS10	Lxvmax	**61.695**	Plaw	62.878	Lxvmax	63.695
DS11	Tlogist	87.275	Txvmin	**84.881**	Tlogist	87.277
DS12	Tlogist	**51.057**	Plaw	52.576	Tlogist	53.057
DS13	Exp	**29.535**	Exp	31.323	Exp	31.956
DS14	Lxvmax	**108.831**	Lxvmax	110.815	Lxvmax	110.831
DS15	Txvmin	**123.265**	Txvmin	124.742	Txvmin	125.265
DS16	Llogist	117.475	Llogist	**115.963**	Llogist	119.475
OSS development projects (group data)						
	NHPP		GBP		GPP	
	Best Model	AIC	Best Model	AIC	Best Model	AIC
DS17	Lxvmax	665.276	Lxvmax	**662.558**	Lxvmax	667.280
DS18	Gtlogist	451.112	Txvmax	**450.308**	Gtlogist	451.541
DS19	Txvmin	**329.766**	Txvmin	330.390	Txvmin	331.759
DS20	Gtlogist	**540.588**	Gtlogist	542.021	Plaw	541.604



(a) DS1.

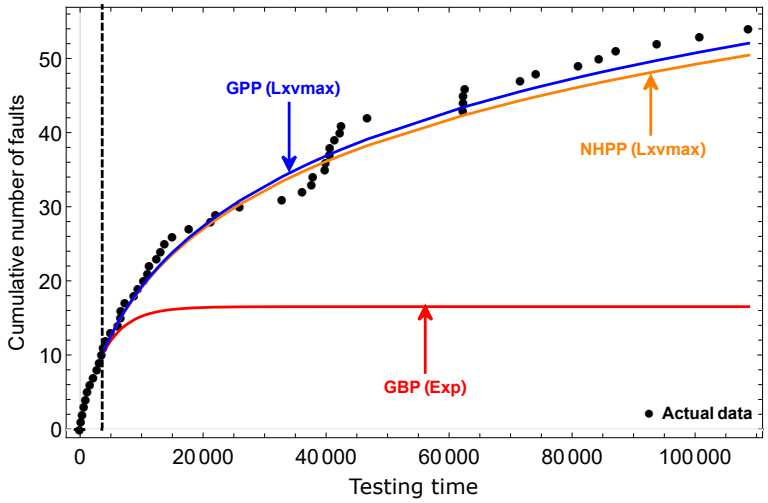


(b) DS9.

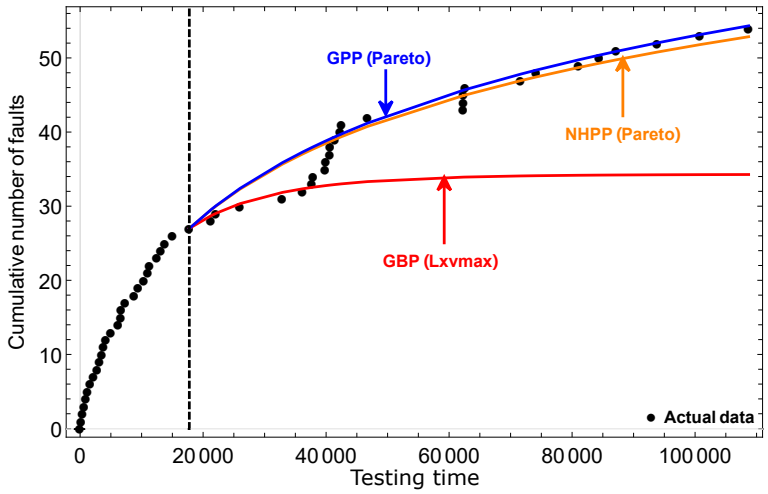


(c) DS17.

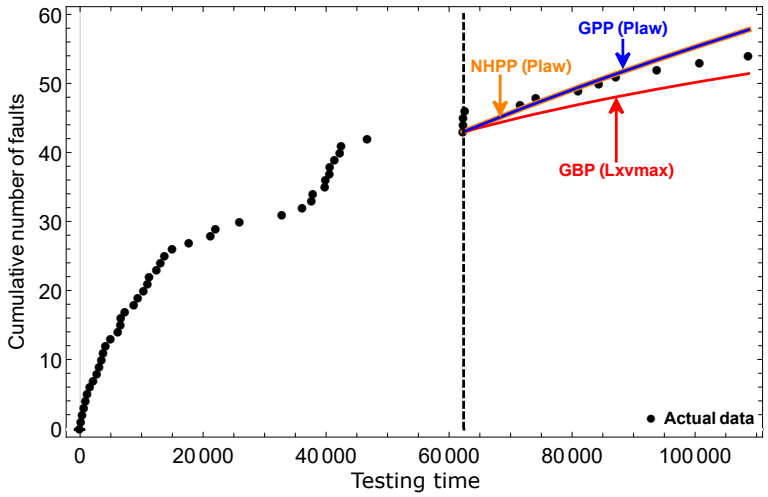
Figure 6.3: Behavior of the cumulative number of software faults with CSS development projects and OSS development project.



(a) 20% observation point.

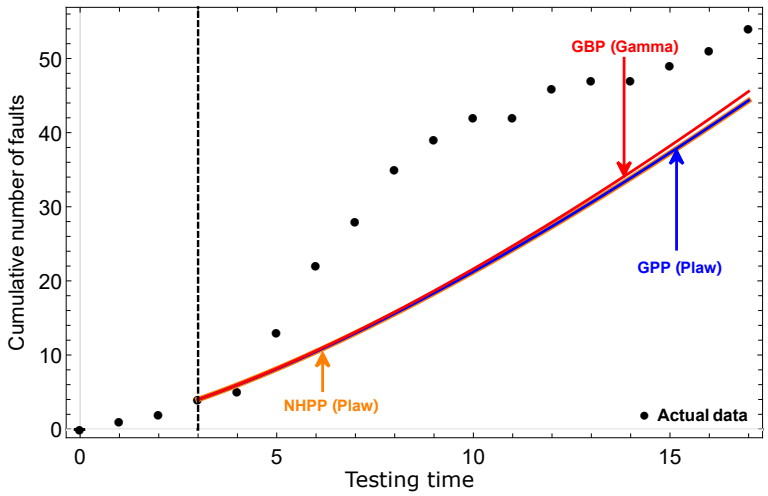


(b) 50% observation point.

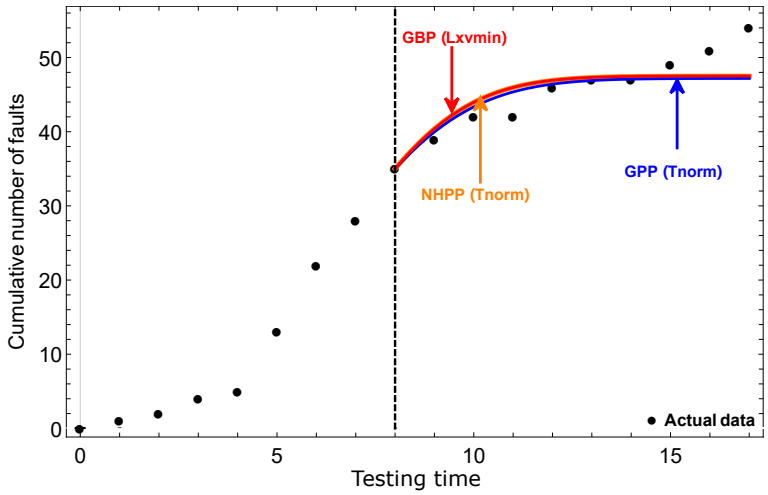


(c) 80% observation point.

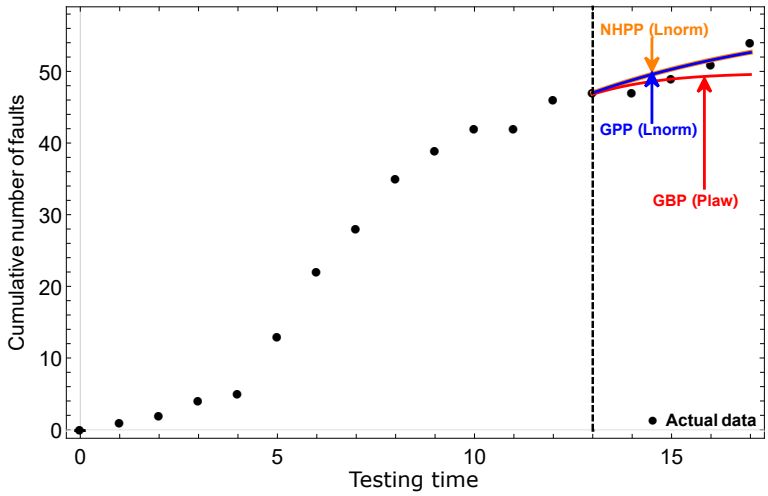
Figure 6.4: Behavior of the predicted cumulative number of software faults in DS1.



(a) 20% observation point.

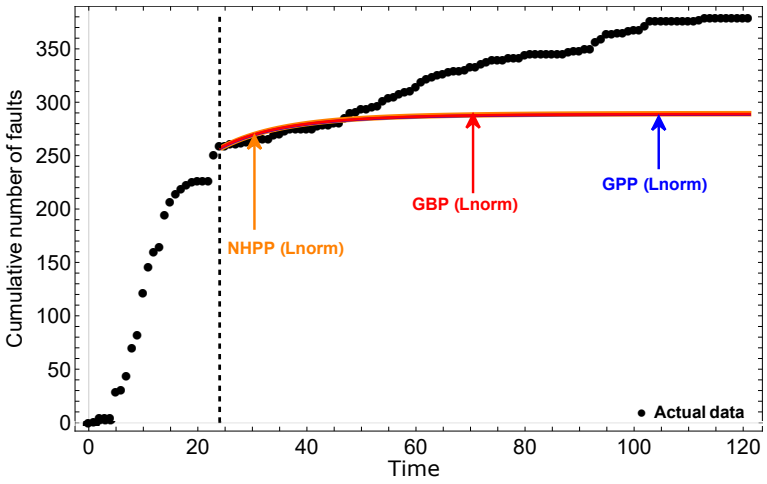


(b) 50% observation point.

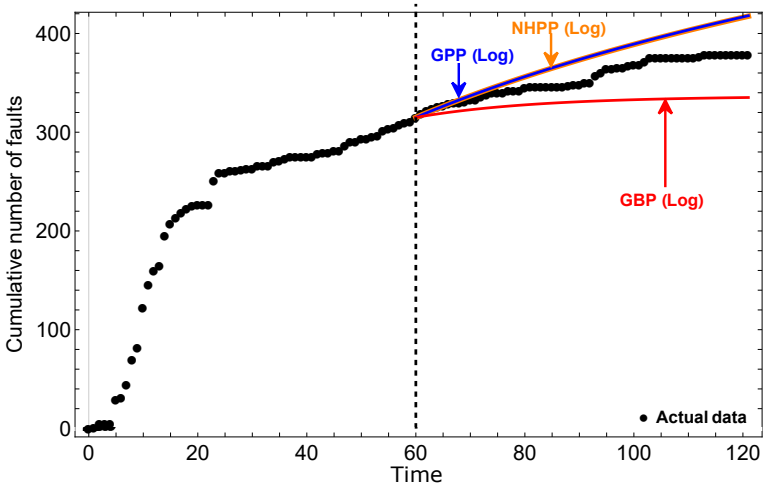


(c) 80% observation point.

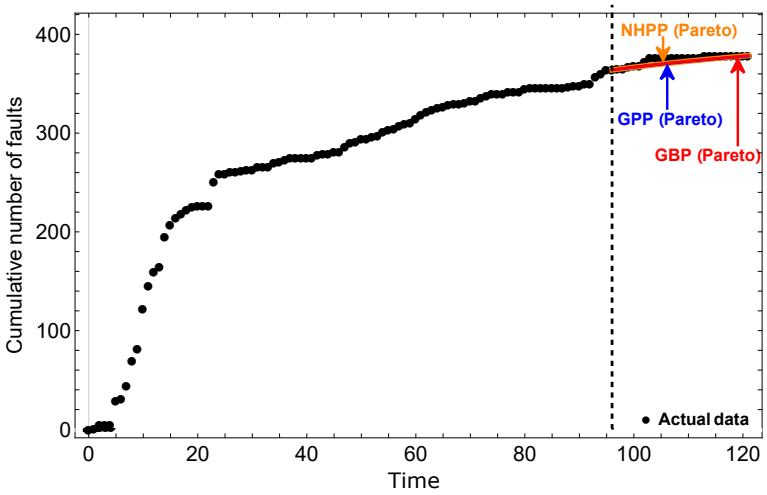
Figure 6.5: Behavior of the predicted cumulative number of software faults in DS9.



(a) 20% observation point.



(b) 50% observation point.



(c) 80% observation point.

Figure 6.6: Behavior of the predicted cumulative number of software faults in DS17.

Table 6.3: Predictive performance based on PMSE with with CSS development projects (time-domain data).

20% Observation Point						
	NHPP		GBP		GPP	
	Best Model	PMSE	Best Model	PMSE	Best Model	PMSE
DS1	Lxvmax	0.323	Exp	3.051	Lxvmax	**0.252**
DS2	Log	0.743	Pareto	**0.729**	Log	0.742
DS3	Lxvmax	**1.371**	Log	1.574	Lxvmax	1.592
DS4	Lxvmax	**1.973**	Lxvmax	2.778	Lxvmax	**1.973**
DS5	Llogist	0.389	Tnorm	**0.354**	Llogist	0.385
DS6	Lxvmax	**1.361**	Lxvmax	1.640	Lxvmax	1.978
DS7	Lnorm	6.270	Pareto	**0.618**	Lxvmax	5.495
DS8	Llogist	**7.756**	Lxvmax	8.995	Llogist	**7.756**
50% Observation Point						
	NHPP		GBP		GPP	
	Best Model	PMSE	Best Model	PMSE	Best Model	PMSE
DS1	Pareto	**0.406**	Lxvmax	1.984	Pareto	0.428
DS2	Txvmin	**0.930**	Tnorm	1.085	Txvmin	0.933
DS3	Log	13.937	Txvmin	**5.167**	Exp	15.115
DS4	Gtlogist	1.581	Lnorm	**1.231**	Llogist	1.338
DS5	Log	1.743	Log	**1.131**	Exp	2.153
DS6	Log	0.383	Log	0.389	Pareto	**0.359**
DS7	Log	0.350	Pareto	3.183	Log	**0.349**
DS8	Tnorm	5.764	Tlogist	**2.877**	Tnorm	5.729
80% Observation Point						
	NHPP		GBP		GPP	
	Best Model	PMSE	Best Model	PMSE	Best Model	PMSE
DS1	Plaw	**0.511**	Lxvmax	0.667	Plaw	**0.511**
DS2	Log	1.239	Log	**1.234**	Exp	1.243
DS3	Plaw	**0.621**	Tlogist	0.807	Plaw	**0.621**
DS4	Lxvmax	**0.655**	Log	0.352	Lxvmax	**0.655**
DS5	Log	0.682	Pareto	0.684	Log	**0.681**
DS6	Lxvmax	0.563	Log	1.028	Lxvmax	**0.560**
DS7	Lxvmax	**0.529**	Pareto	0.567	Lxvmax	**0.529**
DS8	Lxvmax	**0.401**	Txvmax	1.741	Lxvmax	**0.401**

Table 6.4: Predictive performance based on PMSE with CSS development projects (group data).

20% Observation Point						
	NHPP		GBP		GPP	
	Best Model	PMSE	Best Model	PMSE	Best Model	PMSE
DS9	Plaw	3.689	Gamma	3.686	Plaw	**3.572**
DS10	Lxvmax	**1.441**	Lxvmax	1.595	Gamma	1.728
DS11	Tlogist	3.343	Tlogist	3.177	Tlogist	**2.754**
DS12	Exp	3.436	Log	3.436	Exp	**3.210**
DS13	Pareto	0.432	Log	**0.432**	Exp	**0.427**
DS14	Tlogist	2.340	Log	**1.847**	Lxvmax	4.487
DS15	Plaw	0.957	Plaw	3.871	Plaw	**0.945**
DS16	Txvmin	4.032	Tlogist	**3.532**	Tnorm	3.687
50% Observation Point						
	NHPP		GBP		GPP	
	Best Model	PMSE	Best Model	PMSE	Best Model	PMSE
DS9	Tnorm	0.893	Lxvmin	0.921	Tnorm	**0.891**
DS10	Plaw	0.638	Tlogist	2.047	Plaw	**0.635**
DS11	Lxvmax	6.832	Lxvmax	**6.100**	Log	10.777
DS12	Exp	3.522	Exp	**3.443**	Log	4.207
DS13	Exp	0.194	Pareto	**0.193**	Pareto	**0.193**
DS14	Log	**2.047**	Log	5.719	Plaw	2.223
DS15	Lxvmax	1.096	Lxvmax	1.117	Lxvmax	**1.081**
DS16	Txvmin	1.131	Lxvmin	**1.022**	Txvmin	1.133
80% Observation Point						
	NHPP		GBP		GPP	
	Best Model	PMSE	Best Model	PMSE	Best Model	PMSE
DS9	Lnorm	**0.531**	Plaw	1.071	Lnorm	**0.531**
DS10	Exp	0.295	Tlogist	0.295	Tlogist	**0.294**
DS11	Tnorm	0.230	Gtlogist	0.645	Tnorm	**0.228**
DS12	Tnorm	**0.589**	Tlogist	0.781	Tnorm	**0.589**
DS13	Plaw	**0.169**	Llogist	**0.169**	Gtlogist	**0.169**
DS14	Txvmax	**0.741**	Lxvmax	0.943	Lxvmax	0.945
DS15	Txvmin	0.818	Tlogist	**0.143**	Txvmin	0.817
DS16	Lxvmax	0.325	Lxvmax	0.325	Lxvmax	**0.314**

Table 6.5: Predictive performance based on PMSE with OSS development projects (group data).

20% Observation Point						
	NHPP		GBP		GPP	
	Best Model	PMSE	Best Model	PMSE	Best Model	PMSE
DS17	Lnorm	3.978	Lnorm	3.977	Lnorm	**3.961**
DS18	Gtlogist	**12.661**	Log	24.459	Log	21.657
DS19	Plaw	9.353	Gamma	9.333	Log	**4.672**
DS20	Plaw	13.436	Lxvmax	17.963	Plaw	**13.415**
50% Observation Point						
	NHPP		GBP		GPP	
	Best Model	PMSE	Best Model	PMSE	Best Model	PMSE
DS17	Log	**2.594**	Log	3.763	Log	2.630
DS18	Lxvmin	1.756	Lxvmin	**1.030**	Lxvmin	3.711
DS19	Gtlogist	**16.528**	Gtlogist	16.563	Gtlogist	16.558
DS20	Gtlogist	2.004	Gtlogist	1.974	Plaw	**1.759**
80% Observation Point						
	NHPP		GBP		GPP	
	Best Model	PMSE	Best Model	PMSE	Best Model	PMSE
DS17	Pareto	0.698	Pareto	0.782	Pareto	**0.696**
DS18	Tlogist	**0.207**	Tnorm	**0.207**	Tnorm	0.463
DS19	Gamma	**5.346**	Lxvmin	5.387	Llogist	5.378
DS20	Plaw	**2.428**	Lxvmin	**2.428**	Gamma	2.436

Table 6.6: Predictive performance based on the baseline model with minimum AIC in CSS development projects (20% time-domain data).

	NHPP		GBP		GPP	
	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE
DS1	**141.591** (Log)	1.577	142.559 (Plaw)	3.599	143.591 (Log)	**1.575**
DS2	**35.327** (Plaw)	2554.640	37.055 (Plaw)	**1.408**	37.327 (Plaw)	1.091E+10
DS3	**187.583** (Plaw)	4.817	189.398 (Plaw)	**1.769**	189.583 (Plaw)	4.791
DS4	**291.964** (Llogist)	**5.375**	292.373 (Txvmax)	5.624	293.965 (Llogist)	**5.375**
DS5	**1046.910** (Plaw)	0.464	1048.895 (Plaw)	**0.427**	1048.910 (Plaw)	0.470
DS6	**311.745** (Plaw)	**3.573**	313.358 (Plaw)	4.116	313.745 (Plaw)	3.576
DS7	**191.169** (Plaw)	11.723	193.027 (Plaw)	**2.485**	193.169 (Plaw)	11.805
DS8	**752.937** (Txvmax)	**9.350**	753.232 (Txvmax)	9.784	754.937 (Txvmax)	**9.350**

Table 6.7: Predictive performance based on the baseline model with minimum AIC in CSS development projects (50% time-domain data).

	NHPP		GBP		GPP	
	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE
DS1	**403.240** (Log)	**0.471**	403.655 (Plaw)	2.369	405.240 (Log)	**0.471**
DS2	**123.241** (Plaw)	**0.990**	124.923 (Plaw)	1.366	125.241 (Plaw)	**0.990**
DS3	**445.124** (Plaw)	33.395	447.172 (Tlogist)	**5.389**	446.531 (Plaw)	35801.400
DS4	**764.771** (Lxvmax)	4.111	766.231 (Plaw)	**3.311**	766.771 (Lxvmax)	4.113
DS5	**2601.980** (Plaw)	2.834	2604.140 (Plaw)	**2.418**	2603.510 (Plaw)	5.771
DS6	**859.945** (Plaw)	3.884	861.832 (Plaw)	**2.052**	861.945 (Plaw)	3.886
DS7	527.080 (Exp)	**2.993**	**525.815** (Plaw)	3.800	529.068 (Exp)	3.069
DS8	**1900.550** (Plaw)	89.228	1901.390 (Plaw)	**9.748**	1902.550 (Plaw)	89.200

Table 6.8: Predictive performance based on the baseline model with minimum AIC in CSS development projects (80% time-domain data).

	NHPP		GBP		GPP	
	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE
DS1	**690.114** (Log)	**0.702**	690.889 (Plaw)	1.37	692.114 (Log)	**0.702**
DS2	**176.266** (Tlogist)	**8.489**	178.264 (Tlogist)	8.607	177.889 (Txvmin)	14.386
DS3	769.639 (Log)	0.829	**768.020** (Tlogist)	**0.807**	771.641 (Log)	0.831
DS4	**1372.520** (Lxvmax)	**0.655**	1374.290 (Lxvmax)	0.751	1374.520 (Lxvmax)	**0.655**
DS5	**4203.050** (Exp)	**0.683**	4204.420 (Plaw)	0.698	4205.050 (Exp)	**0.683**
DS6	1478.500 (Llogist)	**0.943**	**1477.230** (Plaw)	1.343	1479.313 (Tlogist)	2.368
DS7	**920.101** (Log)	**0.536**	922.031 (Log)	0.608	922.101 (Log)	**0.536**
DS8	3465.000 (Llogist)	3.212	**3449.830** (Txvmax)	**1.741**	3467.000 (Llogist)	3.212

Table 6.9: Predictive performance based on the baseline model with minimum AIC in CSS development projects (20% group data).

	NHPP		GBP		GPP	
	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE
DS9	**10.807** (Plaw)	3.689	12.637 (Plaw)	6.167	12.807 (Plaw)	**3.572**
DS10	**10.864** (Plaw)	**2.145**	12.378 (Plaw)	2.219	12.865 (Plaw)	327.513
DS11	**17.199** (Plaw)	6.764	19.031 (Plaw)	12.026	19.199 (Plaw)	**6.758**
DS12	10.649 (Plaw)	8.017	**10.629** (Exp)	11.14	12.649 (Plaw)	**7.981**
DS13	**6.000** (Plaw)	2.415E+11	7.976 (Plaw)	**7.979**	8.000 (Plaw)	3.606E+6325M
DS14	**20.660** (Lnorm)	34.922	21.781 (Tlogist)	5.540	22.660 (Lnorm)	**5.146**
DS15	**16.959** (Txvmin)	6.600	18.953 (Txvmin)	**6.599**	18.959 (Txvmin)	6.601
DS16	**6.709** (Plaw)	2.081E+13	8.696 (Plaw)	**3.369**	8.730 (Plaw)	2.825E+4245M

Table 6.10: Predictive performance based on the baseline model with minimum AIC in CSS development projects (50% group data).

	NHPP		GBP		GPP	
	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE
DS9	**34.645** (Tlogist)	**1.437**	33.667 (Txvmin)	3.531	36.649 (Tlogist)	**1.437**
DS10	31.051 (Lxvmax)	**2.633**	**29.471** (Plaw)	3.455	33.048 (Lxvmax)	**2.633**
DS11	49.161 (Gtlogist)	20.583	51.099 (Gtlogist)	**18.316**	**49.107** (Plaw)	20.559
DS12	**30.560** (Tlogist)	22.973	33.087 (Plaw)	**7.684**	31.685 (Plaw)	20.962
DS13	**16.878** (Plaw)	0.796	19.911 (Plaw)	**0.732**	18.882 (Plaw)	0.770
DS14	**40.521** (Lxvmax)	5.643	40.768 (Lxvmax)	5.813	42.521 (Lxvmax)	**5.642**
DS15	**70.521** (Plaw)	1.790	72.332 (Plaw)	**1.051**	72.521 (Plaw)	1.768
DS16	65.713 (Txvmin)	**1.131**	**60.151** (Gtlogist)	11.712	67.127 (Txvmin)	1.133

Table 6.11: Predictive performance based on the baseline model with minimum AIC in CSS development projects (80% group data).

	NHPP		GBP		GPP	
	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE
DS9	55.069 (Tlogist)	**1.406**	**53.232** (Txvmax)	1.651	57.069 (Tlogist)	18.845
DS10	**52.523** (Lxvmax)	**0.417**	54.251 (Plaw)	0.655	54.526 (Lxvmax)	0.419
DS11	75.292 (Txvmin)	**0.887**	**73.881** (Tlogist)	1.485	77.293 (Txvmin)	**0.887**
DS12	42.548 (Txvmin)	0.828	**40.788** (Tnorm)	1.242	44.548 (Txvmin)	**0.827**
DS13	**24.272** (Exp)	0.286	25.495 (Exp)	0.664	26.272 (Gtlogist)	**0.169**
DS14	**96.179** (Lxvmax)	0.945	98.179 (Lxvmax)	**0.943**	98.172 (Lxvmax)	0.945
DS15	**111.458** (Plaw)	4.137	112.304 (Tlogist)	**0.143**	113.458 (Plaw)	4.137
DS16	100.326 (Tlogist)	**0.855**	**99.291** (Tlogist)	1.022	102.326 (Tlogist)	**0.855**

Table 6.12: Predictive performance based on the baseline model with minimum AIC in OSS development projects (group data).

20% Observation Point						
	NHPP		GBP		GPP	
	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE
DS17	**274.813** (Llogist)	5.318	275.993 (Llogist)	5.457	276.813 (Llogist)	**5.317**
DS18	**66.096** (Exp)	23.276	66.984 (Plaw)	27.710	68.096 (Exp)	**23.221**
DS19	37.998 (Plaw)	9.353	39.999 (Plaw)	9.434	39.996 (Plaw)	**7.937**
DS20	128.249 (Lxvmax)	**17.784**	129.620 (Lxvmax)	17.963	130.249 (Lxvmax)	**17.784**
50% Observation Point						
	NHPP		GBP		GPP	
	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE
DS17	466.733 (Llogist)	4.692	**459.369** (Txvmax)	**3.868**	468.733 (Llogist)	4.692
DS18	256.109 (Txvmin)	10.354	**255.906** (Txvmax)	10.183	257.261 (Txvmin)	**9.715**
DS19	**87.808** (Plaw)	17.761	89.809 (Exp)	17.782	89.809 (Plaw)	**17.726**
DS20	294.681 (Gtlogist)	2.004	296.677 (Gtlogist)	**1.974**	**292.542** (Gtlogist)	6.267
80% Observation Point						
	NHPP		GBP		GPP	
	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE	AIC (Best Model)	PMSE
DS17	606.600 (Lxvmax)	**0.869**	**603.489** (Lxvmax)	1.063	608.610 (Lxvmax)	0.881
DS18	**407.140** (Gtlogist)	**0.495**	407.330 (Txvmax)	0.588	408.607 (Tlogist)	57.194
DS19	**249.980** (Tlogist)	**8.102**	251.978 (Tlogist)	8.160	251.759 (Tlogist)	37.377
DS20	438.472 (Gtlogist)	3.691	440.477 (Gtlogist)	**3.668**	**435.822** (Gtlogist)	7.141

Table 6.13: Software reliability assessment with NHPP-, GBP- and GPP-based SRMs.

(i) CSS development projects (time-domain data).

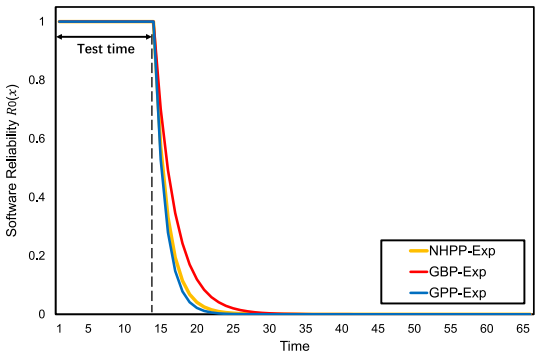
	NHPP		GBP		GPP	
	Best Model	Reliability	Best Model	Reliability	Best Model	Reliability
DS1	Log	**1.092E-17**	Plaw	1.411E-33	Log	1.071E-17
DS2	Plaw	7.081E-138	Tlogist	**1.097E-28**	Plaw	8.886E-138
DS3	Log	4.254E-15	Log	**1.076E-11**	Log	2.098E-15
DS4	Lxvmax	5.567E-15	Lxvmax	1.310E-15	Lxvmax	**6.599E-15**
DS5	Txvmin	**0.000E+00**	Txvmin	**0.000E+00**	Txvmin	**0.000E+00**
DS6	Lxvmin	1.529E-16	Plaw	2.802E-88	Lxvmin	**1.552E-16**
DS7	Log	5.148E-40	Log	**5.681E-40**	Log	1.601E-43
DS8	Lxvmax	**2.939E-48**	Lxvmax	1.193E-51	Lxvmax	2.612E-48

(ii) CSS development projects (group data).

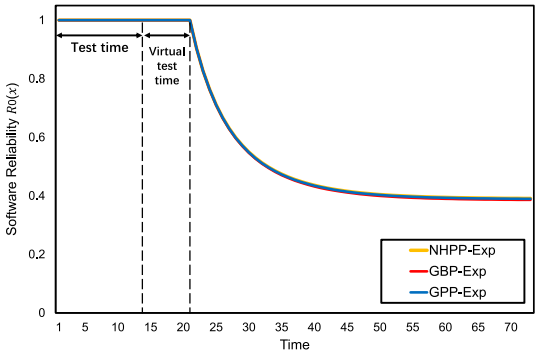
	NHPP		GBP		GPP	
	Best Model	Reliability	Best Model	Reliability	Best Model	Reliability
DS9	Llogist	1.530E-03	Txvmax	**3.876E-02**	Llogist	1.531E-03
DS10	Lxvmax	5.730E-09	Plaw	4.962E-22	Lxvmax	**6.585E-09**
DS11	Tlogist	2.628E-03	Txvmin	5.166E-56	Tlogist	**2.629E-03**
DS12	Tlogist	2.801E-01	Plaw	1.830E-19	Tlogist	**2.808E-01**
DS13	Exp	**5.021E-09**	Exp	4.961E-09	Exp	1.413E-15
DS14	Lxvmax	1.601E-12	Lxvmax	1.052E-12	Lxvmax	**1.613E-12**
DS15	Txvmin	**9.633E-01**	Txvmin	9.610E-01	Txvmin	9.593E-01
DS16	Llogist	6.326E-01	Llogist	**8.012E-01**	Llogist	6.327E-01

(iii) OSS development projects (group data).

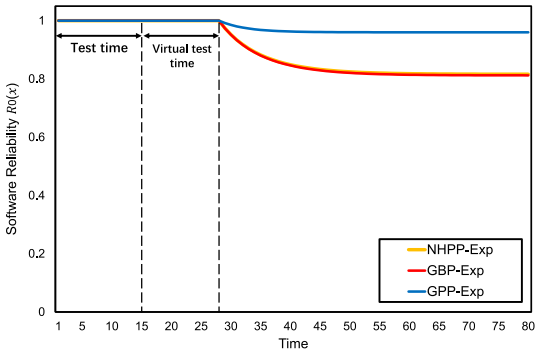
	NHPP		GBP		GPP	
	Best Model	Reliability	Best Model	Reliability	Best Model	Reliability
DS17	Lxvmax	8.578E-03	Lxvmax	**1.209E-02**	Lxvmax	8.497E-03
DS18	Gtlogist	**1.156E-01**	Txvmax	7.438E-02	Gtlogist	8.825E-02
DS19	Txvmin	3.005E-35	Txvmin	1.162E-79	Txvmin	**1.110E-34**
DS20	Gtlogist	4.229E-29	Gtlogist	3.686E-29	Plaw	**8.540E-29**



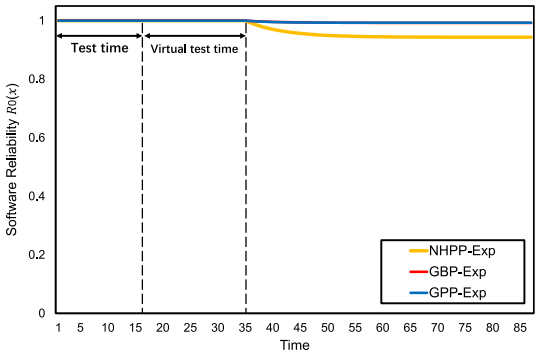
(a) No virtual testing time.



(b) 50% virtual testing time.



(c) 100% virtual testing time.



(d) 150% virtual testing time.

Figure 6.7: Inference of software reliability with virtual testing time (DS13).

Table 6.14: Reliability prediction with virtual testing time.

(i) 50% virtual testing time.

	NHPP		GBP		GPP	
	Best Model	Reliability	Best Model	Reliability	Best Model	Reliability
DS9	Llogist	3.488E-01	Txvmax	**8.727E-01**	Llogist	3.490E-01
DS10	Lxvmax	3.986E-03	Plaw	7.591E-18	Lxvmax	**4.108E-03**
DS11	Tlogist	8.007E-01	Txvmin	**1.000E+00**	Tlogist	8.021E-01
DS12	Tlogist	**9.761E-01**	Plaw	4.137E-38	Tlogist	**9.761E-01**
DS13	Exp	3.909E-01	Exp	3.860E-01	Exp	**4.040E-01**
DS14	Lxvmax	1.417E-04	Lxvmax	**2.546E-01**	Lxvmax	1.495E-04
DS15	Txvmin	**1.000E+00**	Txvmin	**1.000E+00**	Txvmin	**1.000E+00**
DS16	Llogist	9.660E-01	Llogist	9.659E-01	Llogist	**9.663E-01**

(ii) 100% virtual testing time.

	NHPP		GBP		GPP	
	Best Model	Reliability	Best Model	Reliability	Best Model	Reliability
DS9	Llogist	7.237E-01	Txvmax	**9.919E-01**	Llogist	7.255E-01
DS10	Lxvmax	6.004E-02	Plaw	8.753E-12	Lxvmax	**6.234E-02**
DS11	Tlogist	9.890E-01	Txvmin	**1.000E+00**	Tlogist	9.888E-01
DS12	Tlogist	**9.994E-01**	Plaw	1.325E-23	Tlogist	**9.994E-01**
DS13	Exp	8.162E-01	Exp	8.124E-01	Exp	**9.603E-01**
DS14	Lxvmax	9.523E-03	Lxvmax	9.059E-03	Lxvmax	**9.554E-03**
DS15	Txvmin	**1.000E+00**	Txvmin	**1.000E+00**	Txvmin	**1.000E+00**
DS16	Llogist	**9.887E-01**	Llogist	**9.887E-01**	Llogist	**9.887E-01**

(iii) 150% virtual testing time.

	NHPP		GBP		GPP	
	Best Model	Reliability	Best Model	Reliability	Best Model	Reliability
DS9	Llogist	**9.992E-01**	Txvmax	**9.992E-01**	Llogist	8.487E-01
DS10	Lxvmax	1.749E-01	Plaw	7.365E-09	Lxvmax	**1.761E-01**
DS11	Tlogist	9.992E-01	Txvmin	**1.000E+00**	Tlogist	9.992E-01
DS12	Tlogist	**1.000E+00**	Plaw	3.319E-17	Tlogist	**1.000E+00**
DS13	Exp	9.434E-01	Exp	9.922E-01	Exp	**9.930E-01**
DS14	Lxvmax	5.360E-02	Lxvmax	5.319E-02	Lxvmax	**5.376E-02**
DS15	Txvmin	**1.000E+00**	Txvmin	**1.000E+00**	Txvmin	**1.000E+00**
DS16	Llogist	**9.985E-01**	Llogist	**9.985E-01**	Llogist	**9.985E-01**

Table 6.15: Comparison of optimal software release policies with respect to baseline models.

	NHPP		GBP		GPP	
	t_0^* (week)	$C(t_0^*)$	t_0^* (week)	$C(t_0^*)$	t_0^* (week)	$C(t_0^*)$
Exp	38.036	896.408	**32.001**	**752.058**	264.257	7197.580
Gamma	**0.209**	**68.806**	6.094	140.536	5.389	125.808
Pareto	**23.425**	**561.992**	48.679	2912.980	39.723	2011.950
Tnorm	5.459	122.150	5.458	122.208	**5.447**	**121.178**
Lnorm	3.766	208.500	**4.731**	**111.081**	7.201	177.402
Tlogist	4.813	114.309	**4.813**	**114.306**	4.817	114.310
Llogist	7.030	179.993	**4.598**	**108.406**	7.030	179.981
Txvmax	5.538	138.525	**3.691**	**114.149**	3.911	115.186
Lxvmax	0.448	139.218	**4.913**	**114.619**	18.891	740.077
Txvmin	1.133	316.220	4.279	101.959	**4.262**	**101.907**
Lxvmin	4.812	112.380	4.844	112.808	**4.807**	**112.310**
Log	48.676	2983.410	**43.820**	**1085.980**	1.429E-10	6038.370
Plaw	1.429E-10	9746.260	**3.429E-08**	**291.993**	1.429E-10	10809.900
Gtlogist	4.599	109.430	4.602	109.387	**4.598**	**109.278**

Chapter 7

Conclusions

In this thesis, we comprehensively studied the Markov process-based software reliability modeling frameworks. We first complemented the study of some well-known HMP-based SRMs by handling software fault counting group data. Then, We developed dozens of novel parametric and semi-parametric NHPP-based SRMs and proposed two novel software reliability modeling frameworks; GBP and GPP.

In Chapter 2, we have performed the group data analysis for a de-eutrophication SRM based on a pure birth process and compared it with the well-known J&M-SRM in terms of goodness-of-fit and predictive performances. As we have already emphasized, the group data analysis for a de-eutrophication SRM has been left in the software reliability research for a long time. In numerical examples with 8 actual software development project data sets, we have shown that the geometric de-eutrophication SRM was much more attractive to make the software reliability prediction, although the seminal J&M-SRM based on the linear death process has been used more frequently.

In Chapter 3, under the finite-failure and infinite-failure assumptions, we proposed 18 novel type-I NHPP-based SRMs and 26 novel type-II NHPP-based SRMs by considering several representative probability distributions (e.g., generalized exponential distributions family or extreme-value distribution family), Lindley-type distributions and Burr-type distributions as the software fault-detection time c.d.f.s. By analyzing 8 software fault count time-domain data and 8 software fault count group data, we have investigated the goodness-of-fit performance and predictive performance of our SRMs. We have also compared

these SRMs with 11 existing type-I NHPP-based SRMs under the finite-failure assumption. The important point to note is that the type-I and type-II NHPP-based SRMs considered in Chapter 3 have the same software fault detection c.d.f.s, which have never been addressed in the past literature.

The lessons learned from our numerical examples are given in the following:

(i) Our Type-II NHPP-based SRMs could hardly satisfy the goodness-of-fit performance, when compared with the type-I NHPPs.

(ii) Our Type-II NHPP-based SRMs outperformed the existing type-I NHPP-based SRMs for software fault-detection prediction in the early testing phase when group data were available.

(iii) Burr-type NHPP-based SRMs could provide the better goodness-of-fit performances than the other NHPP-based SRMs (including our proposed other NHPP-based SRMs in Chapter 3) in 11 out of 16 data sets.

(iv) Based on PMSE, our Burr-type and Lindley-type NHPP-based SRMs had the better potential for accurate prediction on unknown future fault detection than the existing NHPP-based SRMs in the half of group data sets.

(v) In three observation points of group data sets, our Burr-type NHPP-based SRMs were superior to the existing NHPP-based SRMs in terms of the predictive performance in many cases on the scenario that the best model is selected in terms of the minimum AIC.

(vi) In the software reliability assessment, when we consider goodness-of-fit as the model selection criterion, our type-II NHPP-based SRMs tend to make more conservative predictions than the type-I NHPPs in most cases.

The main contribution of Chapter 3 is to suggest that the Lindley-type and Burr-type NHPP-based SRMs are quite attractive SRMs to describe the software fault-detection processes and should be the possible candidates in selecting the best SRM in terms of goodness-of-fit and predictive performances. Meanwhile, we also confirmed that both finite-failure and infinite-failure assumptions are necessary to be considered in the NHPP-based modeling assumptions. This fact has not been known during the last four decades.

In Chapter 4, we have proposed type-I and type-II local polynomial NHPP-based SRMs, where the software debug rate was given by a local polynomial function. We proposed algorithms to obtain the maximum likelihood estimates of polynomial coefficients in two phases; the estimation phase to investigate the goodness-of-fit and the prediction phase to the inference of an unknown number of software faults in the future. In numerical examples, we have made a comparison of our local polynomial NHPP-based SRMs with the 11 existing type-I and the 3 existing type-II NHPP-based SRMs and confirmed that our type-I local polynomial SRMs could not always provide better goodness-of-fit on AIC, but could outperform the existing NHPP-based SRMs in terms of MSE in almost all cases. On the other hand, in many cases, our local polynomial NHPP-based SRMs outperformed the existing NHPP-based SRMs in terms of predictive performance. From the comprehensive experiments with actual software fault data, our novel NHPP-based SRMs with local polynomial software debug rates are good candidates without determining a specific c.d.f. of the software fault-detection time. However, we have also found that the increase in polynomial degree does not necessarily improve the goodness-of-fit performance of the SRM.

Chapter 5 presented the proportional intensity NHPP-based SRMs (PI-SRMs) with eleven representative baseline intensity functions, which could incorporate multiple time-dependent cumulative/non-cumulative software development/test metrics data. In our numerical experiments with actual software project data, we have quantitatively evaluated the goodness-of-fit and predictive performances of our PI-SRMs and compared them with the common NHPP-based SRMs with the same baseline intensity functions. Finally, we have verified that our SRMs performed well in all data sets and had the excellent potential ability on prediction. By carefully checking the regression coefficients, we have also confirmed that failure identification work was the most important testing metric that could contribute to software debugging, and could improve the goodness-of-fit and predictive performances.

In Chapter 6, we have developed two NHMP-based modeling frameworks to describe the software fault counting processes, where GBP was a binomial type of inverse death process and GPP was a negative binomial type of birth

process. We have derived the log likelihood functions for GBP- and GPP-based SRMs with an arbitrary baseline intensity function for the CSS development projects (time-domain data), CSS development projects (group data) and OSS development projects (group data), and investigated the goodness-of-fit and predictive performances with twenty actual software fault count data sets. The comparison has been made from the viewpoint of difference from the common NHPP-based SRMs with fourteen baseline models. Further, we have discussed how to assess the quantitative software reliability in our NHMP-based SRMs and the software release decision. The lessons learned from numerical experiments are summarized in the following:

(i) Three modeling frameworks; NHPP, GBP and GPP, showed almost similar goodness-of-fit performances for an arbitrary baseline intensity function in CSS development projects (time-domain data), CSS development projects (group data) and OSS development projects (group data) data.

(ii) Our generalized modeling frameworks based on GBP and GPP were superior to the common NHPP-based SRMs in terms of the prediction performance in many cases in the scenario that the best baseline model is selected in terms of the minimum AIC.

(iii) By introducing the virtual testing time, we inferred the quantitative software reliability in a reasonable way, under the assumption that no software fault was detected during the virtual testing time period. It was shown that GPP-based SRMs tend to make optimistic estimations of software reliability.

The approach to generalizing the well-known NHPP-based SRMs was somewhat straightforward, but it has not been done sufficiently during the last five decades. The contribution of Chapter 6 was to describe the software fault count processes with a wide class of Markov processes; GBP and GPP, and to enable the group data analysis. These problems have been left in the software reliability community for a long time. As an interesting insight, it can be pointed out that a great number of baseline models are not needed anymore. Instead, the generalization from the standpoint of the underlying stochastic process would be acceptable to get a more accurate prediction of software faults.

In the future, it is beneficial to implement the de-eutrophication SRM, finite (type-I) and infinite (type-II) Lindley-type, Burr-type, local polynomial NHPP-based SRMs on the well-established software reliability assessment tool. Although SRATS [43] contains 11 well-known NHPP-based SRMs, the main feature is to guarantee the global convergence of model parameters in computing the ML estimates, where the EM (Expectation-Maximization) algorithms are implemented for the respective SRMs. In order to implement reliable and automated ML prediction for the Burr-type NHPP-based SRMs, we need to design the EM algorithms for our proposed SRMs.

Then, we will discuss whether the Lindley-type distribution family can be generalized from the viewpoint of mathematical theory. More specifically, we will try to find whether there exists a relation between the c.d.f. and the p.d.f. to unify the Lindley-type distribution as to the Burr-type distribution family (see Equation (3.29)). If such a differential equation can be derived, then, we may be able to obtain different Lindley-type distributions for describing software fault detection times, as the Burr-type distributions.

Third, we will propose other PI-SRMs with different baseline intensity functions. As we know, some metrics that are more easily observed as time-dependent or non-time-dependent during the testing of software engineering (e.g., the total number of operators, number of program volume, number of lines of comments, number of lines of code, number of lines of executable source code), were missed in the thesis. Therefore, we will continue to propose and investigate novel PI-SRMs in the near future by using the above-mentioned metrics data as well as software fault count data.

Finally, it is attractive to extend GBP- and GPP-based SRMs by introducing nonlinear structures in the transition rates of NHMP, although both GBP and GPP suppose linear structures to represent the state-dependent term. Also, we will develop a numerical inference scheme by solving the Kolmogorov forward equations numerically, without knowing the explicit form of the likelihood function, even when the transition rate in NHMP does not have the decomposition between the state-dependent term and the time-dependent term. It is really a challenging issue to provide the most comprehensive software reliability modeling framework.

Bibliography

[1] M. R. Lyu (ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, 1996.

[2] J. D. Musa, A. Iannino, and K. Okumoto, *Software Reliability, Measurement, Prediction, Application*, McGraw-Hill, New York, 1987.

[3] M. Xie, *Software Reliability Modeling*, World Scientific, Singapore, 1991.

[4] Z. Jelinski, and P. B. Moranda, "Software reliability research," in *Statistical Computer Performance Evaluation*, W. Freiberger (ed.), pp. 465–484, Academic Press, New York, 1972.

[5] P. B. Moranda, "Event-altered rate models for general reliability analysis," *IEEE Transactions on Reliability*, vol. R-28, no. 5, pp. 376–381, 1979.

[6] M. Xie, "On a generalization of the J-M model," in *Proceedings of Reliability 1989*, pp. 5 Ba/3/1-5 Ba/3/7, 1989.

[7] A. A. Abdel-Ghaly, P. Y. Chan, and B. Littlewood, "Evaluation of competing software reliability predictions," *IEEE Transactions on Software Engineering*, vol. SE-12, no. 9, pp. 950–967, 1986.

[8] J. A. Achcar, D. K. Dey, and M. Niverthi, "A Bayesian approach using nonhomogeneous Poisson processes for software reliability models," in *Frontiers in Reliability*, A. P. Basu, K. S. Basu, and S. Mukhopadhyay (eds.), pp. 1–18, World Scientific, Singapore, 1998.

[9] P. Erto, M. Giorgio, and A. Lepore, "The generalized inflection S-shaped software reliability growth model," *IEEE Transactions on Reliability*, vol. 69, no. 1, pp. 228–244, 2020.

[10] A. L. Goel, and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Transactions on Reliability*, vol. R-28, no. 3, pp. 206–211, 1979.

[11] B. Littlewood, "Rationale for a modified Duane model," *IEEE Transactions on Reliability*, vol. R-33, no. 2, pp. 157–159, 1984.

[12] A. L. Goel, "Software reliability models: assumptions, limitations and applicability," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 12, pp. 1411–1423, 1985.

[13] S. S. Gokhale, and K. S. Trivedi, "Log-logistic software reliability growth model," in *Proceedings of the 3rd IEEE International Symposium on High-Assurance Systems Engineering (HASE-1998)*, pp. 34–41, IEEE CPS, 1998.

[14] J. D. Musa, and K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement," in *Proceedings of the 7th International Conference on Software Engineering (ICSE-1984)*, pp. 230–238, ACM, 1984.

[15] M. Ohba, "Inflection S-shaped software reliability growth model," in *Stochastic Models in Reliability Theory*, S. Osaki and Y. Hatoyama (eds.), pp. 144–165, Springer-Verlag, Berlin, 1984.

[16] K. Ohishi, H. Okamura, and T. Dohi, "Gompertz software reliability model: estimation algorithm and empirical validation," *Journal of Systems and Software*, vol. 82, no. 3, pp. 535–543, 3 2009.

[17] H. Okamura, T. Dohi, and S. Osaki, "Software reliability growth models with normal failure time distributions," *Reliability Engineering and System Safety*, vol. 116, no. C, pp. 135–141, 2013.

[18] X. Xiao, "NHPP-based software reliability model with Marshall-Olkin failure time distribution," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E98-A, no. 10, pp. 2060–2068, 2015.

[19] S. Yamada, M. Ohba, and S. Osaki, "S-shaped reliability growth modeling for software error detection," *IEEE Transactions on Reliability*, vol. R-32, no. 5, pp. 475–478, 1983.

[20] M. Zhao, and M. Xie, "On maximum likelihood estimation for a general non-homogeneous Poisson process," *Scandinavian Journal of Statistics*, vol. 23, no. 4, pp. 597–607, 1996.

[21] K. S. Trivedi, and A. Bobbio, *Reliability and Availability Engineering*, Cambridge University Press, Cambridge, UK, 2017.

[22] P. J. Boland, and H. Singh, "A birth-process approach to Moranda's geometric software-reliability model," *IEEE Transactions on Reliability*, vol. 52, no. 2, pp. 168–174, 2003.

[23] L. Kuo, and T. Y. Yang, "Bayesian computation for nonhomogeneous Poisson processes in software reliability," *Journal of the American Statistical Association*, vol. 91, no. 434, pp. 763–773, 1994.

[24] J. D. Musa, "Software reliability data," *Technical Report in Rome Air Development Center* (1979).

[25] A. Wood, "Predicting software reliability," *IEEE Computer*, vol. 20, no. 11, pp. 69–77, 1996.

[26] M. A. Vouk, "Using reliability models during testing with non-operational profile," in *Proceedings of the 2nd Bell-core/Purdue Workshop on Issues in Software Reliability Estimation*, pp. 254-–266, 1992.

[27] H. Okamura, Y. Etani, and T. Dohi, "Quantifying the effectiveness of testing efforts on software fault detection with a logit software reliability growth model," in *Proceedings of 2011 Joint Conference of the 21st International Workshop on Software Measurement (IWSM 2011) and the 6th International Conference on Software Process and Product Measurement (MENSURA-2011)*, pp. 62–68, IEEE CPS, 2011.

[28] https://github.com/OpenEmu/OpenEmu, from Feb. 2012 to Jul. 2022.

[29] https://github.com/vuejs/vue, from Oct. 2013 to Jul. 2022.

[30] https://github.com/flameshot-org/flameshot, from Jun. 2017 to Jul. 2022.

[31] https://github.com/KaTeX/KaTeX, from Aug. 2014 to Jul. 2022.

[32] P. D. Moranda, "Predictions of software reliability during debugging," in *1975 Proceedings of the Annual Reliability and Maintainability Symposium*, pp. 327-332, IEEE CPS, 1975.

[33] P. D. Moranda, "Event-altered rate models for general reliability analysis," *IEEE Transactions on Reliability*, vol. R-28, no. 5, pp. 376–381, 1979.

[34] O. Gaudoin, and J. L. Soler, "Statistical analysis of the geometric deeutrophication software-reliability model," *IEEE Transactions on Reliability*, vol. 41, no. 4, pp. 518–524, 1992.

[35] J.P. Boland, and H. Singh, "A birth-process approach to Moranda's geometric software-reliability model," *IEEE Transactions on Reliability*, vol. 52, no. 2, pp. 168–174, 2003.

[36] T. Vasanthi, and G. Arulmozhi, "Reliability computation of Moranda's geometric software reliability model," *Economic Quality Journal*, vol. 22. no. 2, pp. 261–272, 2007.

[37] N. D. Singpurwalla, and S. P. Wilson, "Software reliability modeling," *International Statistical Review*, vol. 62, pp. 289–317, 1994.

[38] J. G. Shanthikumar, "A general software reliability model for performance prediction," *Microelectronics and Reliability*, vol. 21, no. 5, pp. 671–682, 1981.

[39] M. Xie, "A Markov process model for software reliability analysis," *Applied Stochastic Models and Data Analysis*, vol. 6, no. 4, pp. 207–213, 1990.

[40] D. R. Miller, "Exponential order statistic models of software reliability growth," *IEEE Transactions on Software Engineering*, vol. SE-12, pp. 12–24, 1986.

[41] Y. Lam, *The Geometric Process and Its Applications*, World Scientific, 2007.

[42] Y. L. Gat, *Recurrent Event Modeling based on the Yule Process Application to Water Network Asset Management*, vol. 2, John Wiley & Sons, 2016.

[43] H. Okamura, and T. Dohi, "SRATS: software reliability assessment tool on spreadsheet," in *Proceedings of the 24th International Symposium on Software Reliability Engineering (ISSRE-2013)*, pp. 100–117, IEEE CPS, 2013.

[44] J. Hishitani, S. Yamada, and S. Osaki, "Reliability assessment measures based on software reliability growth model with normalized method," *Journal of Information Processing*, vol. 14, no. 2, pp. 178-–183, 1991.

[45] R.E. Barlow, and F. Proschan, *Mathematical Theory of Reliability*, Wiley, New York, 1965.

[46] D. Cox, and P. Lewis, *The Statistical Analysis of Series of Events*, Springer, Dordrecht, 1966.

[47] E. Cretois, and O. Gaudoin, "New results on goodness-of-fit tests for the power-law process and application to software reliability," *International Journal of Reliability, Quality and Safety Engineering*, vol. 5, no. 3, pp. 249–267, 1998.

[48] J. T. Duane,"Learning curve approach to reliability monitoring," *IEEE Transactions on Aerospace*, vol. 2, no. 2, pp. 563–566, 1964.

[49] Q. Xiao, T. Dohi and H. Okamura, "Lindley type distributions and software reliability assessment," in *Proceedings of The 8th Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modeling (APARM 2020)*, 6 pages, IEEE CPS, 2020.

[50] D. V. Lindley, "Fiducial distributions and Bayes' theorem," *Journal of the Royal Statistical Society B*, vol. 20, no. 1, pp. 102–107, 1958.

[51] M. E. Ghitany, B. Atieh, and S. Nadarajah, "Lindley distribution and its application," *Mathematics and Computers in Simulation*, vol. 78, no. 4, pp. 493–506, 2008.

[52] J. Mazucheli, and J. A. Achcar, "The Lindley distribution applied to competing risks lifetime data," *Computer Methods and Programs in Biomedicine*, vol. 104, vol. 2, pp. 189–192, 2011.

[53] M. E. Ghitany, D. K. Al-Mutairi, N. Balakrishnan, and L. J. Al-Enezi, "Power Lindley distribution and associated inference," *Computational Statistics and Data Analysis*, vol. 64, no. 1, pp. 20–33, 2013.

[54] D. K. Al-Mutairi, M. E. Ghitany, and D. Kundu, "Inference on stress-strength reliability from Lindley distributions," *Communications in Statistics – Theory and Methods*, vol. 42, no. 8, pp. 1443–1463, 2013.

[55] S. Nadarajah, H. S. Bakouch, and R. Tahmasbi, "A generalized Lindley distribution," *Sankhya*, vol. 73, no. 2. pp. 331–359, 2011.

[56] S. K. Ashour, and M. A. Eltehiwy, "Exponentiated power Lindley distribution," *Journal of Advanced Research*, vol. 6, , no. 6, pp. 895–905, 2015.

[57] S. Nedjar and H. Zeghdoudi, "On gamma Lindley distribution: properties and simulations," *Journal of Computational and Applied Mathematics*, vol. 298, no. 1, pp. 167–174, 2016.

[58] M. E. Ghitany, F. Alqallaf, D. K. Al-Mutairi, and H. A. Husain, "A two-parameter weighted Lindley distribution and its applications to survival data," *Mathematics and Computers in Simulation*, vol. 81, no. 6, pp. 1190–1201, 2011.

[59] J. Mazucheli, F. Louzada, and M. E. Ghitany, "Comparison of estimation methods for the parameters of the weighted Lindley distribution," *Applied Mathematics and Computation*, vol. 220, no. 1, pp. 463–471, 2013.

[60] A. A. Baqer, "Gompertz-Lindley distribution and its applications," *Master Thesis in the College of Graduate Studies, Kuwait University*, 2019.

[61] I. W. Burr, "Cumulative frequency functions," *Annals of Mathematical Statistics*, vol. 13, no. 2, pp. 215–232, 1942.

[62] P. R. Tadikamalla, "A look at the Burr and related distributions," *International Statistical Review*, vol. 48, no. 3, pp. 337–344, 1980.

[63] W. J. Zimmer, J. B. Keats and F. K. Wang, "The Burr XII distribution in reliability analysis," *Journal of Quality Technology*, vol. 30. no. 4, pp. 386–394, 2018.

[64] A. A. Abdel-Ghaly, G.R. Al-Dayian, F.H. Al-Kashkari, "The use of Burr-type XII distribution on software reliability growth modelling," *Microelectronics and Reliability*, vol. 37, no. 2, pp. 305–313, 1997.

[65] D. R. Miller, "Exponential order statistic models of software reliability growth," *IEEE Transactions on Software Engineering*, vol. SE-12, pp. 12–24, 1986.

[66] H.-C. Kim, and H.-K. Park, "The comparative study of software optimal release time based on Burr distribution," *International Journal of Advancements in Computing Technology*, vol. 2, no. 3, pp. 119–128, 2010.

[67] H.-C. Kim, "Assessing software reliability based on NHPP using SPC," *International Journal of Software Engineering and Its Applications*, vol. 7, no. 6, pp. 61–70, 2013.

[68] J. H. An, "Two model comparisons of software reliability analysis for Burr-type XII distribution," *Journal of the Korean Data & Information Science Society*, vol. 23, no. 4, pp. 815–823, 2012.

[69] R. S. Prasad, K. V. M. Mohan and G. Sridevi, "Burr-type XII software reliability growth model," *International Journal of Computer Applications*, vol. 108, no. 16, pp. 16–20, 2014.

[70] R. S. Prasad, K. V. M. Mohan and G. Sridevi, "Monitoring Burr-type XII software quality using SPC," *International Journal of Applied Engineering Research*, vol. 9, no. 22, pp. 16651–16660, 2014.

[71] R. S. Prasad, B. R. Devi and G. Sridevi, "Assessing Burr-type XII software reliability for interval domain data using SPC," *Computer Engineering*, vol. 79, pp. 30335–30340, 2015.

[72] M. S. Ravikumar, and R. R. L. Kantam, "Software reliability model based on Burr-type XII distribution," *International Journal of Advanced Engineering Research and Applications*, vol. 2, no. 9, pp. 561–564, 2017.

[73] S. F. Islam, "Introducing Burr-type XII testing-effort with change point based software reliability growth model," *Global Scientific Journals*, vol. 8, no. 8, pp. 2712–2718, 2020.

[74] N. Ahmad, M. G. M. Khan, S. M. K. Quadri and M. Kumar, "Modelling and analysis of software reliability with Burr-type III testing-effort and release-time determination," *Journal of Modelling in Management*, vol. 4, no. 1, pp. 28–54, 2009.

[75] N. Ahmad, S. M. K. Quadri, M. G. N. Khan and M. Kumar, "Software reliability growth models incorporating Burr-type III test-effort and cost-reliability analysis," *International Journal of Computer Science and Information Technologies*, vol. 2, no. pp.555–562, 2011.

[76] K. Sobhana, and R. S. Prasad, "Burr-type III software reliability with SPC - An order statistics approach," *International Journal of Research Studies in Computer Science and Engineering*, vol. 2, no. 3, pp. 21–24, 2015.

[77] C. S. Chowdary, R. S. Prasad, and K. Sobhana, "Burr-type III software reliability growth model," *IOSR Journal of Computer Engineering*, vol. 17, no. 1, pp. 49–54, 2015.

[78] G. Sridevi, and C. M. S. Rani, "Comparison of software reliability analysis for Burr distribution," *Journal of Theoretical and Applied Information Technology*, vol. 81, no. 1, pp. 144–150, 2015.

[79] G. Sridevi, and S. Akbar, "Burr-type X software reliability growth model," *Asian Journal of Information Technology*, vol. 15, no. 16, pp.2988–2991, 2016.

[80] H.-C. Kim, "A study on comparative evaluation of software reliability model applying modified exponential distribution," *International Journal of Engineering Research and Technology*, vol. 13, no. 5, pp. 867–872, 2020.

[81] T. Imanaka, and T. Dohi, "Software reliability modeling based on Burr XII distributions," *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, vol. E98-A, no. 10, pp. 2091–2095, 2015.

[82] Y. Zhao, T. Dohi and H. Okamura, "Software Test-Run Reliability Modeling with Non-homogeneous Binomial Processes," in *Proceedings of the 2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC 2018)*, pp. 145-154, 2018.

[83] H. Okamura, T. Hirata, and T. Dohi, "Semi-parametric approach for software reliability evaluation using mixed gamma distributions," *International Journal of Software Engineering and Its Applications*, vol. 7, no. 4, pp. 401–414, 2013.

[84] H. Okamura, and T. Dohi, "Phase-type software reliability model: parameter estimation algorithms with grouped data," *Annals of Operations Research*, vol. 244, no. 1, pp. 177–208, 2016.

[85] M. Nafreen, and L. Fiondella, "A family of software reliability models with bathtub-shaped fault detection rate," *International Journal of Reliability, Quality and Safety Engineering*, vol. 28, no. 05, 2150034, 2021.

[86] S. Yamada and S. Osaki, "An error detection rate theory for software reliability growth models", *Transactions of the Institute of Electronics and Communication Engineers of Japan*, vol. E68, no. 5, pp. 292–296, 1985.

[87] L. J. Bain, "Analysis for the linear failure-rate life-testing distribution," *Technometrics*, vol. 16, no. 4, pp. 551–559, 1974.

[88] N. Balakrishnan, and H. J. Malik, "Order statistics from the linear-exponential distribution, part I: increasing hazard rate case," *Communications in Statistics – Theory and Methods*, vol. 15, no. 1, pp. 179—203, 1986.

[89] M. A. W. Mahmoud and H. SH. Al-Nagar, "On generalized order statistics from linear exponential distribution and its characterization," *Statistical Papers*, vol. 50, pp. 407–418, 2009.

[90] J. F. Lawless, *Statistical Models and Methods for Lifetime Data*, Wiley, NewYork, 1982.

[91] S. A. Krane, "Analysis of survival data by regression techniques," *Technometrics*, vol. 5, no. 2, pp. 161–174, 1963.

[92] V. I. Kogan, "Polynomial models of generalized bathtub curves and related moments of the order statistics," *SIAM Journal of Applied Mathematics*, vol. 48, no. 2, pp. 416–424, 1988.

[93] A. Csenki, "On continuous lifetime distributions with polynomial failure rate with an application in reliability," *Reliability Engineering and System Safety*, vol. 96, pp. 1587–1590, 2011.

[94] D. Bagkavos, and P. N. Patil, "Local polynomial fitting in failure rate estimation," *IEEE Transactions on Reliability*, vol. 57, no. 1, pp. 41–52, 2008.

[95] J. C. Berger, and D. Sun,"Bayesian analysis for the poly-Weibull distribution," *Journal of the American Statistical Association*, vol. 88, no. 422, pp. 1412–1418, 1993.

[96] A. C. Davison, and F. Louzaada-Neto,"Inference for the poly-Weibull model," *Journal of the Royal Statistical Society Series D*, vol. 49, no. 2, pp. 189–196, 2000.

[97] J. F. Freels, D. A. Timme, J. J. Pignatiello, R. L. Warr, and R. R. Hill, "Maximum likelihood estimation for the poly-Weibull distribution," *Quality Engineering*, 2019.

[98] N. Demiris, D. Lunn, and L. D. Sharples,"Survival extrapolation using the poly-Weibull model," *Statiostical Methods in Medical Research*, vol. 24, no. 2, pp. 287–301, 2015.

[99] K. Rinsaka, K. Shibata, and T. Dohi, "Proportional Intensity-Based Software Reliability Modeling with Time-Dependent Metrics," in *proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, vol. 1, pp. 369-376, 2006.

[100] K. Rinsaka, K. Shibata, and T. Dohi, "PISRAT: Proportional Intensity-Based Software Reliability Assessment Tool," in *proceedings of the 13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*, pp. 43-52, 2007.

[101] S. Murphy, and P. Sen, "Time-dependent coefficients in a Cox type regression model," *Stochastic Processes and Their Applications*, vol. 39, no. 1, pp. 153-180, 1991.

[102] L. Tian, D. Zucker and L. J. Wei, "On the Cox model with time-varying regression coefficient," *Journal of the American Statistical Association*, vol. 100, no. 469, pp. 172-183, 2005.

[103] D. R. Cox, "Regression models and life-tables," *Journal of the Royal Statistical Society*, vol. B-34, pp. 197-220, 1972.

[104] J. F. L. Lawless,"Regression methods for Poisson process data," *Journal of the American Statistical Association*, vol. 82, no. 399, pp. 808-815, 1987.

[105] D. G. Kendall, "On the generalized "birth-and-death" process," *Annals of Mathematical Statistics*, vol. 19, pp. 1–5, 1948.

[106] M. Shaked, F. Spizzichino and F. Suter, "Nonhomogeneous birth processes and l_∞ spherical densities, with applications in reliability theory," *Probability in the Engineering and Informational Sciences*, vol. 16, no. 3, pp. 271–288, 2002.

[107] J. G. Shanthikumar, "A general software reliability model for performance prediction," *Microelectronics and Reliability*, vol. 21, no. 5, pp. 671–682, 1981.

[108] M. K. Smotherman, and K. Zemoudeh, "A non-homogeneous Markov model for phased-mission reliability analysis," *IEEE Transactions on Reliability*, vol. 38, no. 5, pp. 585–590, 1989.

[109] M. K. Smotherman, and R. M. Geist, "Phased mission effectiveness using a nonhomogeneous Markov reward model," *Reliability Engineering and System Safety*, vol. 27, pp. 241–255, 1990.

[110] G. Cosulich, P. Firpo, and S. Savio, "Power electronics reliability impact on service dependability for railway systems: A real case study," in *Proceedings of IEEE International Symposium on Industrial Electronics (ISIE-1996)*, pp. 996–1001, IEEE CPS, 1998.

[111] V. P. Koutras, A. N. Platis, and G. A. Gravvanis, "Optimal server resource reservation policies for priority classes of users under cyclic non-homogeneous markov modeling," *European Jopurnal of Operational Research*, vol. 198, pp. 545–556, 2009.

[112] V. P. Koutrasa, A. N. Platisa, and G. A. Gravvanis, "On the optimization of free resources using non-homogeneous Markov chain software rejuvenation models," *Reliability Engineering and System Safety*, vol. 92, pp. 1724–1732, 2007.

[113] S. S. Gokhale, T. Philip, and P. N. Marinos, "A non-homogeneous Markov software reliability model with imperfect repair," in *Proceedings of the 12th Annual Conference on Computer Assurance (COMPASS-1997)*, pp. 262–270, IEEE CPS, 1997.

[114] S. S. Gokhale, P. N. Marinos, and K. S. Trivedi, "Effect of repair policies on software reliability," in *Proceedings of IEEE International Computer Performance and Dependability Symposium (ICPDS-1997)*, pp. 105–116, IEEE CPS, 1997.

[115] S. S. Gokhale, M. R. Lyu, and K. S. Trivedi, "Analysis of software fault removal policies using a non-homogeneous continuous time Markov chain," *Software Quality Journal*, vol. 12, pp. 211–2301, 2004.

[116] G. J. Schick, and R. W. Wolverton, "Assessment of software reliability," in *Proceedings of the Operations Research (the 11th Annual Meeting of the German Operations Research Society)*, pp. 395–422, Physica-Verlag, Wurzburg, Wien, 1973.

[117] G. J. Schick, and R. W. Wolverton, "An analysis of competing software reliability models," *IEEE Transactions on Software Engineering*, vol. SE-4, no. 2, pp. 104–120, 1978.

[118] W. L. Wagoner, "The final report on a software reliability measurement study," *Aerospace Corporation Report*, vol. TOR-0074 (4112)-1, 1973.

[119] B. Littlewood, "Theories of software reliability: How good are they and how can they be improved?," *IEEE Transactions on Software Engineering*, vol. SE-6, no. 5, pp. 489–500, 1980.

[120] B. Littlewood, "Stochastic reliability growth: A model for fault removal in computer-program and hardware-designs," *IEEE Transactions on Reliability*, vol. R-30, no. 4, pp. 313–320, 1981.

[121] J. G. Shanthikumar, "Software reliability models: A review," *Microelectronics and Reliability*, vol. 23, no. 5, pp. 903–943, 1983.

[122] H. Konno, "On the exact solution of a generalized Polya process," *Advances in Mathematical Physics*, vol. 2010, no.2010(2010), pp. 1–12, 2010.

[123] H. Okamura and T. Dohi, "Software reliability modeling based on mixed Poisson distributions," *International Journal of Reliability, Quality and Safety Engineering*, vol. 15, no. 1, pp. 9–32, 2008.

[124] J. H. Cha, "Characterization of the generalized Polya process and its applications," *Journal of Applied Probability*, vol. 46, no. 3, pp. 1148–1171, 2014.

[125] A. W. Marshall, and I. Olkin, "A new method for adding a parameter to a family of distributions with application to the exponential and Weibull families," *Biometrika*, vol. 84, no. 3, pp. 641–652, 1997.

[126] Y. L. Gat, *Recurrent Event Modeling based on the Yule Process Application to Water Network Asset Management*, vol. 2, John Wiley & Sons, London, 2016.

[127] Z. G. Asfaw, and B. H. Lindqvist, "Extending minimal repair models for repairable systems: A comparison of dynamic and heterogeneous extensions of a nonhomogeneous Poisson process," *Reliability Engineering and System Safety*, vol. 140, pp. 53–58, 2015.

[128] S. R. Dalal, and A. A. McIntosh, "When to stop testing for large software systems with changing code," *IEEE Transactions on Software Engineering*, vol. 20, no. 4, pp. 318–323, 1994.

[129] H. S. Koch, and P. Kubat, "Optimal release time for computer software," *IEEE Transactions on Software Engineering*, vol. SE-9, no. 3, pp. 323-327, 1983.

[130] K. Okumoto, and L. Goel, "Optimum release time for software systems based on reliability and cost criteria," *Journal of Systems and Software*, vol. 1, pp. 315–318, 1980.

[131] S. Yamada, and S. Osaki, "Cost-reliability optimal release policies for software systems," *IEEE Transactions on Reliability*, vol. R-34, no. 5, pp. 422–424, 1985.

[132] B. Yang, H. Hu, and L. Jia, "A study on uncertainty in software cost and its impact on optimal software release time," *IEEE Transactions on Software Engineering*, vol. 34, no. 6, pp. 813–835, 2008.

Publication List of the Author

[1] S. Li, T. Dohi, and H. Okamura, "Software reliability analysis via geometric de-eutrophication models with group data," *International Journal of Systems Assurance Engineering and Management*, pp. 1-9, 2022.

[2] S. Li, T. Dohi, and H. Okamura, "Burr-type NHPP-based software reliability models and their applications with two type of fault count data," *Journal of Systems and Software*, vol. 191, no. 3, 15 pages, 2022.

[3] S. Li, T. Dohi, and H. Okamura, "Local polynomial software reliability models and their application," *Journal of Reliability and Statistical Studies*, accepted for publication, (Accepted date: February 3, 2022).

[4] S. Li, T. Dohi, and H. Okamura, "A comprehensive analysis of proportional intensity-based software reliability models with covariates," *Electronics*, vol. 11, no. 15, 18 pages, 2022.

[5] S. Li, T. Dohi, and H. Okamura, "Are Infinite-failure NHPP-based Software Reliability Models Useful?," *Software*, vol. 2, no. 1, pp. 1-18, 2022.

[6] S. Li, T. Dohi, and H. Okamura, "Non-homogeneous Markov Process Modeling for Software Reliability Assessment," *Transactions on Reliability*, accepted for publication, (Accepted date: December 30, 2022).

[7] S. Li, "A useful parametric family to characterize NHPP-based software reliability models," in *Proceedings of the 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, pp. 23-24, 2021.

[8] S. Li, T. Dohi, and H. Okamura, "An NHPP-based software reliability model with local polynomial debug rate," in *Proceedings of the Reliability and Maintenance Engineering Summit 2021 (RMES 2021)*, pp. 98-105, 2021.

[9] S. Li, T. Dohi, and H. Okamura, "A comprehensive evaluation for Burr-type NHPP-based software reliability models," in *Proceedings of the 8th International Conference on Dependable Systems and Their Applications (DSA)*, pp. 1-11, 2021 (Awarded as the best paper in DSA 2021).

[10] S. Li, T. Dohi, and H. Okamura, "Infinite NHPP-based software reliability models with Burr-type distributions," in *Proceedings of the 2022 Asia Pacific International Symposium on Advanced Reliability and Maintenance Modeling (APARM 2022)*, 5 pages, (Accepted date: July 8, 2022; Taipei, Taiwan), 2022.

[11] S. Li, T. Dohi, and H. Okamura, "A Summary of NHPP-based Software Reliability Modeling with Lindley-type Distributions," in *System Reliability and Security: Techniques and Methodologies*, CRC Press, Taylor and Francis Group, (Accepted date: November 16, 2022).