software

MDPI

Article

Are Infinite-Failure NHPP-Based Software Reliability Models Useful?

Siqiao Li, Tadashi Dohi * and Hiroyuki Okamura

Graduate School of Advanced Science and Engineering, Hiroshima University, Hiroshima 739-8511, Japan
* Correspondence: dohi@hiroshima-u.ac.jp

Abstract: In the literature, infinite-failure software reliability models (SRMs), such as Musa-Okumoto SRM (1984), have been demonstrated to be effective in quantitatively characterizing software testing processes and assessing software reliability. This paper primarily focuses on the infinite-failure (type-II) non-homogeneous Poisson process (NHPP)-based SRMs and evaluates the performances of these SRMs comprehensively by comparing with the existing finite-failure (type-I) NHPP-based SRMs. In more specific terms, to describe the software fault-detection time distribution, we postulate 11 representative probability distribution functions that can be categorized into the generalized exponential distribution family and the extreme-value distribution family. Then, we compare the goodness-of-fit and predictive performances with the associated 11 type-I and type-II NHPP-based SRMs. In numerical experiments, we analyze software fault-count data, collected from 16 actual development projects, which are commonly known in the software industry as fault-count time-domain data and fault-count time-interval data (group data). The maximum likelihood method is utilized to estimate the model parameters in both NHPP-based SRMs. In a comparison of the type-I with the type-II, it is shown that the type-II NHPP-based SRMs could exhibit better predictive performance than the existing type-I NHPP-based SRMs, especially in the early stage of software testing.

Keywords: software reliability; infinite-failure models; non-homogeneous Poisson processes; maximum likelihood estimation; goodness-of-fit performance; predictive performance

Citation: Li, S.; Dohi, T.; Okamura, H. Are Infinite-Failure NHPP-Based Software Reliability Models Useful? *Software* **2023**, *2*, 1–18. https://doi.org/10.3390/software2010001

Academic Editor: Tommi Mikkonen

Received: 18 November 2022
Revised: 19 December 2022
Accepted: 20 December 2022
Published: 23 December 2022

1. Introduction

In actual software development projects, clients frequently expect software products to be of high quality. As software products tend to become more complex with increasing size, software reliability is gradually gaining much attention from developers as an important attribute of software quality. Therefore, in the modern software development process, developers concentrate their human and material resources on the testing process to detect and fix inherent faults as much as possible for the purpose of improving software reliability. Due to the high cost of software testing, quantification of software reliability is also regarded as a significant concern during the verification phase.

To the best of our knowledge, the probabilistic behavior of the fault detection and correction process during the software testing phase in software reliability engineering is usually characterized by any stochastic counting process. On the other hand, after the software product is released, the probability of a product not experiencing any failure caused by software faults over a specific time interval is usually defined as quantitative software reliability. To measure the above software reliability, over the last five decades, hundreds of probabilistic models known as software reliability models (SRMs) have been implemented to quantitatively assess software reliability during the testing and operational phases. Among the existing SRMs, the non-homogeneous Poisson process (NHPP)-based SRMs are recognized as a very important class because of their mathematical tractability and high applicability. By modeling the software failure time, Kuo and Yang [1] classified NHPP-based SRMs into general order statistics SRMs and record value statistics SRMs. The

same authors [1] proposed an alternative and more general classification by dividing NHPP-based SRMs into two types; finite-failure (type-I) and infinite-failure (type-II) NHPP-based SRMs with mean value functions, which are defined as the expected cumulative number of software failures. The best-known finite-failure (type-I) NHPP-based SRM was proposed by Goel and Okumoto [2], who assumed the exponential distribution as the fault-detection time distribution in software testing. The mean value function there is in proportion to the cumulative distribution function (CDF) of the exponential distribution. After that, postulating the other fault-detection time distributions, several type-I NHPP-based SRMs were proposed in the literature, such as the truncated-normal NHPP-based SRM [3], the log-normal NHPP-based SRM [3,4], the truncated-logistic NHPP-based SRM [5], the log-logistic NHPP-based SRM [6], the extreme-value NHPP-based SRMs [7,8], the gamma NHPP-based SRM [9,10], and the Pareto NHPP-based SRM [11]. At the same time, introducing a series of commonly used lifetime CDFs for modeling software failure time in reliability engineering, a few infinite-failure (type-II) NHPP-based SRMs have also been developed and widely used to quantitatively evaluate software reliability. The power-law process model [12–14] and the logarithmic Poisson execution time model [15,16] are classified as type-II NHPP-based SRMs. Note that the use of type-I NHPP-based SRMs does not necessarily imply that all inherent software faults are detected over an infinite time horizon. In other words, the precise number of inherent faults cannot be known, even if software testing is performed for an indefinite period. Thus, the *finiteness* in the type-I NHPP-based SRMs holds in the sense of the expectation of a cumulative number of detected faults. Unfortunately, a fair comparison between the type-I and type-II NHPP-based SRMs has not been made in past, because only a limited number of type-II NHPP-based SRMs have been considered in the literature.

Our research question of this paper is "Are infinite-failure NHPP-based SRMs useful?" More specifically, this paper investigates whether infinite-failure NHPP-based SRMs can guarantee better goodness-of-fit performance for the fault-count data collected in the software testing phase in comparison with finite-failure NHPP-based SRMS, and whether they can guarantee more accurate performance in predicting the number of software faults. Goodness-of-fit and predictive performance are generally recognized as the critical factors determining which SRM should be practically applied to quantitatively assess software reliability.

The original contribution of this paper is to investigate the type-II NHPP-based SRMs with the representative 11 CDFs in the literature. Three of these type-II NHPP-based SRMs are confirmed to be equivalent to the existing Cox-Lewis process, logarithmic Poisson execution time model, and power-law process, while the remaining eight type-II SRMs are novel SRMs. We confirm that the corresponding type-I and type-II NHPP-based SRMs can be obtained by importing the same software fault-detection time distribution CDFs to the finite- and infinite-failure assumptions, respectively. We make a comprehensive comparison between the existing type-I NHPP-based SRMs and their associated type-II NHPP-based SRMs. As shown in [17], it seems enough to consider 11 kinds of software fault-detection time CDFs in making goodness-of-fit and predictive performance comparisons between two different NHPP-based modeling frameworks.

The rest of this paper is organized as follows. Section 2 describes the definition of NHPP and illustrates NHPP-based software reliability modeling under the finite-failure and the infinite-failure hypotheses. We present 11 existing type-I NHPP-based SRMs based on the finite-failure hypothesis in [17] and propose 11 type-II NHPP-based SRMs with the same CDFs. The maximum likelihood approach to estimate the model parameters is summarized. We confirm that maximum likelihood estimation can be used for parameter estimation of our type-II NHPP-based SRMs for software fault-count time-domain data. We also give specific expressions of likelihood function and log-likelihood function for software fault-count time-interval data (group data), which are more common in the industry.

In the numerical examples in Section 3, we employ a total of 16 datasets collected from 16 actual development projects. In each dataset, we investigate the goodness-of-fit

performance and predictive performance of type-I NHPP-based SRMs and type-II NHPP-based SRMs that have completed parameter estimation by maximum likelihood estimation. In addition, we also use these SRMs to quantitatively evaluate software reliability over a given time period, and analyze the applicability of the type-I NHPPs as well as the type-II NHPPs in predicting software reliability. In Section 4, the paper is summarized with some remarks and future directions.

2. Non-Homogeneous Poisson processes

2.1. Preliminary

As a well-known Markov process, a non-homogeneous Poisson process (NHPP) is regarded as a generalization of the classical homogeneous Poisson process (HPP). If the intensity at time point t in the definition of HPP is given by a function $\breve{}(t)$ with respect to t, then an HPP can be generalized to 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 not on the past history of the process, which is also known as the Markov property.
- The 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 $\Pr\{N(t + \Delta t) - N(t) = 1\} = o(\Delta t) + \breve{}(t)\Delta t$. $\breve{}(t)$ is an absolutely continuous function, and is named the *intensity function* of NHPP. 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.
- As a typical Markov process, the Kolmogorov forward equations of NHPP can be written as

$$\frac{d}{dt}P_0(t) = -\breve{}(t;\theta)P_0(t), \tag{1}$$

$$\frac{d}{dt}P_n(t) = \breve{}(t;\boldsymbol{\theta})P_{n-1}(t) - \breve{}(t;\boldsymbol{\theta})P_n(t), \qquad n = 1, 2, \cdots, \tag{2}$$

with $P_0(0) = 1$ and $P_n(0) = 0$, where $\boldsymbol{\theta}$ represents the free parameter vector in the transition rate function $\breve{}(t;\boldsymbol{\theta})$. By solving the above simultaneous equations, the steady-state transition probability $\Pr\{N(t) = n | N(0) = 0\} = P_n(t)$ is given by

$$P_n(t) = \exp(-M(t;\boldsymbol{\theta}))\frac{\{M(t;\boldsymbol{\theta})\}^n}{n!}, \quad n = 0, 1, 2, \ldots. \tag{3}$$

Through the Poisson property, $M(t;\boldsymbol{\theta})$ is defined as the *mean value function* of NHPP and represents the expected cumulative number of event occurrences during the interval $(0, t]$.

2.2. NHPP-Based SRMs

Most textbooks [16,18,19] 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 NHPP-based SRMs* and *infinite-failure NHPP-based SRMs*.

2.2.1. Finite-Failure (Type-I) NHPP-Based SRMs

In the software reliability modeling framework developed based on finite-failure (type-I) NHPP, before testing, the remaining number of software faults is assumed to obey a Poisson distribution with a positive mean μ_0. Each software fault is assumed to be detected at independent and identically distributed (i. i. d.) random times, and is fixed immediately

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 cumulative distribution function (CDF). $\boldsymbol{\alpha}$ indicates the free parameter vector in the CDF. Then, a binomial distributed random variable with probability $F(t;\boldsymbol{\alpha})$ with a Poisson distributed population with parameter μ_0 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

$$M(t;\boldsymbol{\theta}) = \mu_0 F(t;\boldsymbol{\alpha}), \tag{4}$$

which can also be recognized as the cumulative number of faults detected by the software testing at time point t with $\boldsymbol{\theta} = (\mu_0, \boldsymbol{\alpha})$ and $\lim_{t\to\infty} M(t;\boldsymbol{\theta}) = \mu_0$ (> 0). This property is consistent with the assumption of software reliability modeling for type-I NHPP that the number of initial remaining faults expected before software testing begins is finite. In Table 1, we summarize 11 existing type-I NHPP-based SRMs with their associated CDFs and bounded mean value functions, which were employed in the software reliability assessment tool on the spreadsheet (SRATS) by Okamura and Dohi [17].

Table 1. The representative existing finite-failure (type-I) NHPP-based SRMs.

SRM & Time Distribution	$F(t;\boldsymbol{\alpha})$	$M(t;\boldsymbol{\theta})$
Exp [2] (Exponential distribution)	$1-\exp(-\mu_1 t)$	$\mu_0 F(t;\boldsymbol{\alpha})$
Gamma [9,10] (Gamma distribution)	$\int_0^t \frac{\mu_2^{\mu_1} s^{\mu_2-1}\exp(-\mu_2 s)}{\Gamma(\mu_1)} ds$	$\mu_0 F(t;\boldsymbol{\alpha})$
Pareto [11] (Pareto distribution)	$1-\left(\frac{\mu_1}{t+\mu_1}\right)^{\mu_2}$	$\mu_0 F(t;\boldsymbol{\alpha})$
Tnorm [3] (Truncated normal distribution)	$\frac{1}{\sqrt{2\pi\mu_1}} \int_{-\infty}^t \exp\left(-\frac{(s-\mu_2)^2}{2\mu_1^2}\right) ds$	$\mu_0 \frac{F(t;\boldsymbol{\alpha})-F(0;\boldsymbol{\alpha})}{1-F(0;\boldsymbol{\alpha})}$
Tlogist [5] (Truncated logistic distribution)	$\frac{1-\exp(-\mu_1 t)}{1+\mu_2\exp(-\mu_2 t)}$	$\mu_0 \frac{F(t;\boldsymbol{\alpha})-F(0;\boldsymbol{\alpha})}{1-F(0;\boldsymbol{\alpha})}$
Txvmax [8] (Truncated extreme-value maximum distribution)	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$\mu_0 \frac{F(t;\boldsymbol{\alpha})-F(0;\boldsymbol{\alpha})}{1-F(0;\boldsymbol{\alpha})}$
Txvmin [8] (Truncated extreme-value minimum distribution)	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$\mu_0 \frac{F(0;\boldsymbol{\alpha})-F(t;\boldsymbol{\alpha})}{F(0;\boldsymbol{\alpha})}$
Lnorm [3,4] (Log-normal distribution)	$\frac{1}{\sqrt{2\pi\mu_1}} \int_{-\infty}^t \exp\left(-\frac{(s-\mu_2)^2}{2\mu_1^2}\right) ds$	$\mu_0 F(\ln\ t;\boldsymbol{\alpha})$
Llogist [6] (Log-logistic distribution)	$\frac{1-\exp(-\mu_1 t)}{1+\mu_2\exp(-\mu_2 t)}$	$\mu_0 F(\ln\ t;\boldsymbol{\alpha})$
Lxvmax [8] (Log-extreme-value maximum distribution)	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$\mu_0 F(\ln\ t;\boldsymbol{\alpha})$
Lxvmin [7] (Log-extreme-value minimum distribution)	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$\mu_0(1-F(-\ln t;\boldsymbol{\alpha}))$

$(\mu_0 > 0, \mu_1 > 0, \mu_2 > 0)$; $\Gamma(\cdot)$: standard gamma function; erfc$(\cdot)$: complementary error function; ln$(\cdot)$: natural logarithmic function.

Even though the type-I NHPP-based SRMs are recognized as plausible models in term of software reliability growth phenomena, it must be acknowledged that reliability engineers sometimes feel discomfort when handling finite-failure NHPPs, since the inter-failure time distributions in the type-I NHPP-based SRMs are *defective* [20]. Let us suppose that the random variables T_1, $T_2, \ldots, T_n$ represent the first, second, $\ldots$, n-th failure times

that occur after the software testing starts at $T_0 = 0$. Let the random variables $X_1, X_2, \ldots, X_n$ denote the inter-failure times between two consecutive failures:

$$T_n = \sum_{j=1}^{n} X_j = T_{n-1} + X_n, \quad n = 0, 1, 2, \ldots. \tag{5}$$

From Equations (3) and (5), the CDF of T_n can be obtained as

$$\begin{aligned} G_n(t;\boldsymbol{\theta}) &= P\{T_n \le t \text{ (the } n-\text{th failure occurs up to } t)\} \\ &= P\{N(t) \ge n \text{ (at least } n \text{ failures occur before time } t)\} \\ &= \int_0^t \frac{\breve{}(x;\boldsymbol{\theta})[M(x;\boldsymbol{\theta})]^{n-1}}{(n-1)!} \exp(-M(x;\boldsymbol{\theta}))dx \\ &= \sum_{j=n}^{\infty} \frac{[M(t;\boldsymbol{\theta})]^j}{j!} \exp(-M(t;\boldsymbol{\theta})) \\ &= 1 - \sum_{j=0}^{n-1} \frac{[M(t;\boldsymbol{\theta})]^j}{j!} \exp(-M(t;\boldsymbol{\theta})). \end{aligned} \tag{6}$$

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 CDF of T_n is defective. Similarly, for realizations of T_i $(i = 1, 2, \ldots, n)$, t_1, $t_2, \ldots, t_n$, we can obtain the CDF of the inter-failure time X_n in the time interval $(t_{n-1},\ t_{n-1} + x)$ as follows.

$$F_n(x;\ \boldsymbol{\theta}) = 1 - \Pr\{\ N(t_{n-1} + x) - N(t_{n-1}) = 0 | N(t_{n-1}) = n-1\} = 1 - \exp(-(M(t_{n-1} + x;\boldsymbol{\theta}) - M(t_{n-1};\boldsymbol{\theta}))), \tag{7}$$

where $\Pr\{\ N(t_{n-1} + x) - N(t_{n-1}) = 0 | N(t_{n-1}) = n-1\}$ 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., $\lim_{t\to\infty} M(t;\boldsymbol{\theta}) = \mu_0$, when x is infinite, Equation (7) can be reduced to $1 - e^{-(\mu_0 - M(t_{n-1};\boldsymbol{\theta}))} < 1$. 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 CDF of type-I NHPP is also defective. For the type-I NHPP-based SRMs, it is not meaningful to discuss some reliability metrics, such as mean time between failures (MTBF), because the finite moments of T_n and X_n always diverge.

2.2.2. Infinite-Failure (Type-II) NHPP-Based SRMs

Type-II NHPP assumes that a new software fault is not inserted at each software debugging. However, this assumption may be somewhat specific, because 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. Similarly to the classical preventive maintenance modeling [21], if each software failure is minimally repaired through the debugging, the mean value function of software fault-detection process is unbounded and is given by

$$M(t;\boldsymbol{\alpha}) = -\ln(1 - F(t;\boldsymbol{\alpha})), \tag{8}$$

where $\lim_{t\to\infty} M(t;\boldsymbol{\alpha}) \to \infty$. It is obvious that the CDFs, $G_n(t;\boldsymbol{\theta})$ and $F_n(x;\boldsymbol{\theta})$ in Equations (6) and (7) are not defective; for instance, $\lim_{t\to\infty} G_n(t;\boldsymbol{\theta}) = 1$ 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 [22] is one of earliest type-II NHPPs. The unbounded mean value function of this model is given by $M(t;\boldsymbol{\alpha}) = (\exp(\mu_1 + \mu_2 t) - \exp(\mu_1))/\mu_2$ 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 [17].

Another well-known type-II NHPP-based SRM is referred to as a power-law process model [12–14], where mean value function and CDF 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 [17]. Besides the above two representative NHPPs, the well-known logarithmic Poisson execution time SRM [15,16] 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 [17]. In Table 1, it is easy to see that the same CDFs are used for type-I Txvmin SRM v.s. type-II Cox-Lewis SRM, type-I Lxvmin SRM v.s. type-II Power-law SRM, and type-I Pareto SRM v.s. type-II Musa-Okumoto SRM, respectively. Hence, by substituting 11 software fault-detection time CDFs in Table 1 into Equation (8), we can derive the corresponding type-II NHPP-based SRMs in Table 2.

Table 2. Infinite-failure (type-II) NHPP-based SRMs.

SRM & Time Distribution	$F(t;\boldsymbol{\alpha})$	$M(t;\boldsymbol{\theta})$
Exp (HPP) (Exponential distribution)	$1-\exp(-\mu_1 t)$	$\mu_1 t$
Gamma (Gamma distribution)	$\int_0^t \frac{\mu_2^{\mu_1} s^{\mu_2-1}\exp(-\mu_2 s)}{\Gamma(\mu_1)} ds$	$\ln(\Gamma(\mu_1)) - \ln\left(\Gamma\left(\mu_1, \frac{t}{\mu_2}\right)\right)$
Pareto (Musa-Okumoto) [15,16] (Pareto distribution)	$1-\left(\frac{\mu_1}{t+\mu_1}\right)^{\mu_2}$	$-\mu_2(\ln(\mu_1) - \ln(\mu_1+t))$
Tnorm (Truncated normal distribution)	$\frac{1}{\sqrt{2\pi\mu_1}}\int_{-\infty}^{t}\exp\left(-\frac{(s-\mu_2)^2}{2\mu_1^2}\right)ds$	$\ln\left(\mathrm{erf}\left(\frac{\mu_2}{\sqrt{2\mu_1}}\right)+1\right) -$ $\ln\left(\mathrm{erf}\left(\frac{\mu_2-t}{\sqrt{2\mu_1}}\right)+1\right)$
Tlogist (Truncated logistic distribution)	$\frac{1-\exp(-\mu_1 t)}{1+\mu_2\exp(-\mu_2 t)}$	$\ln(\exp(\mu_2/\mu_1)+\exp(t/\mu_1)) -$ $\ln(\exp(\mu_2/\mu_1)+1)$
Txvmax (Truncated extreme-value maximum distribution)	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$\ln(1-\exp(-\exp(\mu_2/\mu_1)))$ $-\ln\left(1-\exp\left(-\exp\left(\frac{\mu_2-t}{\mu_1}\right)\right)\right)$
Cox-Lewis [22] (Truncated extreme-value minimum distribution)	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$-\ln(\exp(-\exp(\mu_2/\mu_1)(\exp(t/\mu_1)-1)))$
Lnorm (Log-normal distribution)	$\frac{1}{\sqrt{2\pi\mu_1}}\int_{-\infty}^{t}\exp\left(-\frac{(s-\mu_2)^2}{2\mu_1^2}\right)ds$	$\ln(2) - \ln\left(\mathrm{erf}\left(\frac{\mu_2-\ln(t)}{\sqrt{2}\mu_1}\right)+1\right)$
Llogist (Log-logistic distribution)	$\frac{1-\exp(-\mu_1 t)}{1+\mu_2\exp(-\mu_2 t)}$	$\ln\left(\exp(\mu_2/\mu_1)+t^{1/\mu_1}\right) - \mu_2/\mu_1$
Lxvmax (Log-extreme-value maximum distribution)	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$-\ln\left(1-\exp\left(-\exp\left(\frac{\mu_2-\ln(t)}{\mu_1}\right)\right)\right)$
Power-law [12–14] (Log-extreme-value minimum distribution)	$\exp\left(-\exp\left(-\frac{t-\mu_2}{\mu_1}\right)\right)$	$\mu_2/\mu_1 t^{1/\mu_1}$

2.3. Parameter Estimation

For the existing type-I and type-II NHPP-based SRMs, the maximum likelihood (ML) estimation is a typical technique that is widely applied in software reliability modeling. The parameters of the maximized log-likelihood function (LLF) provide the ML estimates. In addition, the LLF depends on the observed fault-count data, the intensity function and/or the mean value function in the type-I and type-II NHPP-based SRMs. Next, we give the likelihood functions for the software fault-count time-domain data and the software fault-count time-interval data (group data).

2.3.1. Software Fault-Count Time-Domain Data

Consider that the total number of faults observed in the testing phase is m_D before the time observation point t_{m_D}, where the time sequence consisting of the time points at which each fault is detected is given by $\boldsymbol{D} = \{t_1, t_2, \ldots, t_{m_D}\}$. 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, for the time-domain data, the likelihood function of NHPP is as follows.

$$L(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}; \boldsymbol{D}) = \exp(-M(t_{m_D}; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})) \prod_{i=1}^{m_D} \lambda(t_i; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}). \tag{9}$$

Taking logarithm of both sides in Equation (9), the log-likelihood function is obtained as

$$\ln L(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}; \boldsymbol{D}) = \sum_{i=1}^{m_D} \ln \lambda(t_i; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) - M(t_{m_D}; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}). \tag{10}$$

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

2.3.2. Software Fault-Count Time-Interval Data (Group Data)

Software fault-count time-interval data (group data) consist of the number of detected faults in a set of calendar-time-based time intervals $(t_{i-1}, t_i]$ $(i = 1, 2, \ldots, m_I)$. Each record in these types of data is composed of the observation time point t_i and n_i, the cumulative number of faults detected in the time interval $(0, \ t_i]$, written as (t_i, n_i). Calendar time is usually measured as testing days or weeks. When the group data $\boldsymbol{I} = \{(t_i, n_i), i = 1, 2, \ldots, m_I\}$ are available, the likelihood function and LLF are given by

$$\begin{aligned} L(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}; \boldsymbol{I}) = - \prod_{i=1}^{m_I} & \exp[M(t_i; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) \\ & - M(t_{i-1}; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})] \left[\frac{[M(t_i; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) - M(t_{i-1}; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})]^{n_i - n_{i-1}}}{(n_i - n_{i-1})!} \right] \end{aligned} \tag{11}$$

and

$$\begin{aligned} \ln L(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}; \boldsymbol{I}) = & \sum_{i=1}^{m_I} (n_i - n_{i-1}) \ln\{M(t_i; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) - M(t_{i-1}; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})\} \\ & - \sum_{i=1}^{m_I} \ln\{(n_i - n_{i-1})!\} - M(t_{m_I}; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}), \end{aligned} \tag{12}$$

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

3. Performance Comparison

3.1. Datasets

In the selection of data sources for numerical experiments, we selected the well-known benchmark software fault-count datasets in software reliability engineering, which are observed in mission-critical systems. Although the evolution of these systems may be slower than that of business-oriented systems, the effects of a failure are much greater. Hence, reliability is particularly important for the developers of these mission-critical systems. In the industry, the software fault-count data observed in the distributed test environment for mission-critical systems can be divided into two categories: software fault-count time-domain data and software fault-count time-interval data (group data). We selected eight sets of each type of data, which have been widely utilized as fault-count data in software reliability engineering. The details of these datasets are shown in Tables 3 and 4, respectively.

Table 3. Software fault-count time-domain datasets.

	Data Source	Nature of System	Testing Length (CPU Time)	Numbers of Detected Faults
TDDS1	SYS2 [23]	Real-time command and control system	108708	54
TDDS2	S10 [23]	Real-time command and control system	233700	38
TDDS3	SYS3 [23]	Military application	67362	38
TDDS4	S27 [23]	Single-user workstation	4312598	41
TDDS5	SYS4 [23]	Operating system	52422	53
TDDS6	Project J5 [18]	Real-time command and control system	5090	73
TDDS7	S17 [23]	Single-user workstation	19572126	101
TDDS8	SYS1 [23]	Single-user workstation	88682	136

Table 4. Software fault-count time-interval datasets (group data).

	Data Source	Nature of System	Testing Length (Week)	Numbers of Detected Faults
TIDS1	SYS2 [23]	Real-time command and control system	17	54
TIDS2	NASA-supported project [24]	Inertial navigating system	14	9
TIDS3	SYS3 [23]	Military application	14	38
TIDS4	DS3 [25]	Embedded application for printer	30	52
TIDS5	DS2 [25]	Embedded application for printer	33	58
TIDS6	Release 3 [26]	Tandem software system	12	61
TIDS7	DS1 [25]	Embedded application for printer	20	66
TIDS8	Release 2 [26]	Tandem software system	19	120

3.2. Goodness-of-Fit Performance

Assuming that the parameters of the SRMs were estimated by maximum likelihood estimation, in the first experiment, we employed two criteria for evaluating the goodness-of-fit performance of the 11 type-I and type-II NHPP-based SRMs, for instance, Akaike information criterion (AIC):

$$\mathrm{AIC}\left(\hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}\right) = 2 \times (\text{number of free parameters}) - 2\ \ln L\left(\hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}\right) \tag{13}$$

and the mean squared error (MSE);

$$\mathrm{MSE}\left(\hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}; \boldsymbol{D}\right) = \frac{\sum_{i=1}^{m_D}\left(i - M\left(t_i; \hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}\right)\right)^2}{m_D} \tag{14}$$

or

$$\mathrm{MSE}\left(\hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}; \boldsymbol{I}\right) = \frac{\sum_{i=1}^{m_I}\left(n_i - M\left(t_i; \hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}\right)\right)^2}{m_I}, \tag{15}$$

respectively. In Equations (14) and (15), n_i is defined as the count of detected faults in the time interval $(0,\ t_i]$, m_D and m_I are the lengths of time-domain and group data, and $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\alpha}}$ are the ML estimates by maximizing $\ln L(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}; \boldsymbol{D})$ and $\ln L(\boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}; \boldsymbol{I})$. The AIC with ML estimates generally represented an approximation of the Kullback–Leibler divergence between our proposed SRM and the empirical stochastic process behind the fault-count data, while direct MSE exhibited a vertical distance between the estimated mean value function and the fault-count data. A smaller AIC/MSE indicated that the SRM had a better goodness-of-fit performance (showing a better fit to the underlying data).

Figures 1 and 2 plot the behavior of the mean value functions of type-I and type-II SRMs in the time-domain data, TDDS1, and the group data, TIDS7. 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 AIC, respectively. Not surprisingly, the two modeling frameworks showed slightly different growth trends. More specifically, the type-I (orange curve) always fitted better to the actual data in the tail segment, in both the time-domain and the group data. However, we still were not able to make a comprehensive assessment regarding which SRM exhibited a better fitting ability over the whole dataset. It was therefore necessary to think about AIC as well as MSE as such criteria. First, in Table 5, we make a more precise comparison between our proposed type-II and the existing type-I on AIC and MSE. Without comparing them with each other, it is evident that in the vast majority of cases, the best models among the type-I SRMs were given by the extreme-value distributions. By contrast, the type-II Pareto (Musa-Okumoto) SRM performed better than the other SRMs. In the next step, by comparing the best type-I and type-II SRMs for each dataset, it is not difficult to observe that in three cases (TDDS1, TDDS3, and TDDS6), the type-II SRMs provided a smaller AIC than the type-I SRMs. However, in all the datasets, the type-I SRMs provided a smaller MSE than the type-II SRMs.

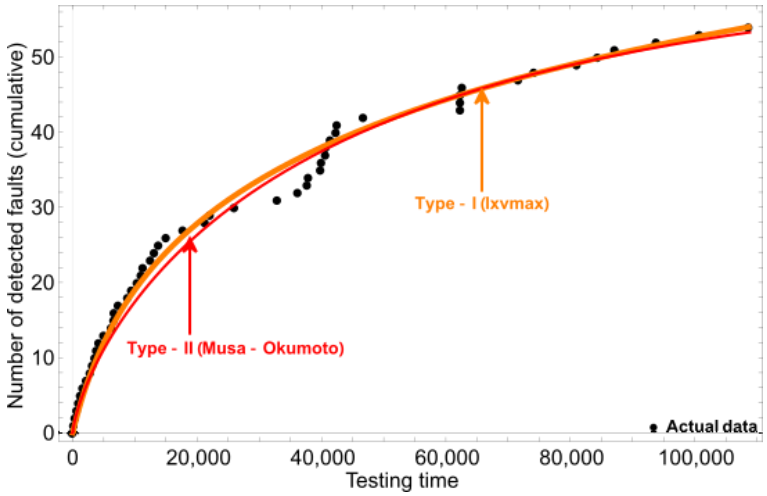

Figure 1. Behavior of mean value functions in TDDS1.

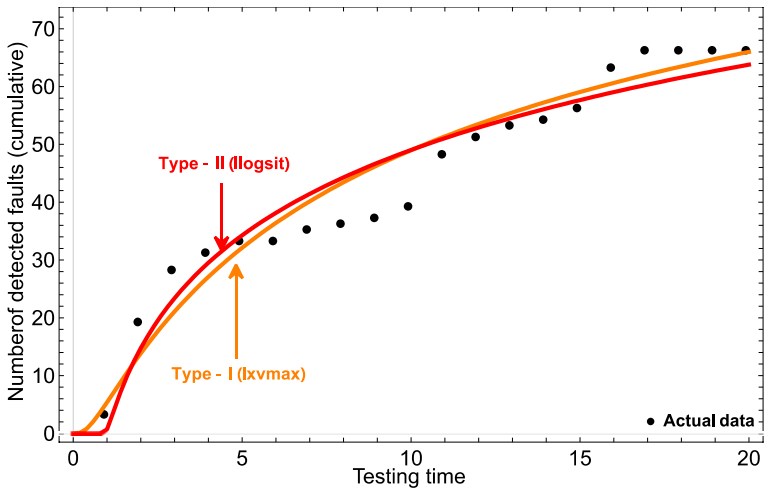

Figure 2. Behavior of mean value functions in TIDS7.

In Table 6, we compared the SRMs of our type-II NHPP with the existing type-I NHPP-based SRMs in eight group datasets. It can be seen that our type-II SRMs could guarantee a

smaller AIC than the existing type-I in three cases (TIDS2, TIDS3, and TIDS7), but at the same time, it still could not outperform the type-I from the viewpoint of MSE for any group dataset. 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 time-domain data, the three existing type-II NHPP-based SRMs, Musa-Okumoto, Cox-Lewis, and power-law SRMs, could indicate the better experimental results.

Table 5. Goodness-of-fit results in time-domain data.

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

Table 6. Goodness-of-fit results in group data.

	Type-I NHPP			**Type-II NHPP**		
	Best SRM	**AIC**	**MSE**	**Best SRM**	**AIC**	**MSE**
TIDS1	Llogist	73.053	4.115	Tlogist	85.339	48.269
TIDS2	Exp	29.911	0.118	Exp	27.753	0.186
TIDS3	Lxvmax	61.694	3.239	Llogist	60.674	3.557
TIDS4	Llogist	117.470	9.408	Llogist	148.438	45.178
TIDS5	Txvmin	123.265	2.122	Tlogist	138.029	24.847
TIDS6	Tlogist	51.052	1.968	Cox-Lewis	63.556	27.199
TIDS7	Lxvmax	108.831	22.514	Llogist	107.211	24.394
TIDS8	Tnorm	87.267	6.151	Cox-Lewis	91.919	31.232

3.3. Predictive Performance

Notably, according to the previous studies, SRMs with better goodness-of-fit do not necessarily provide an 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 employed the prediction mean squared error (PMSE) to measure the predictive performance of our type-II SRMs, where

$$\mathrm{PMSE}\left(\hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}; \boldsymbol{D}\right) = \frac{\sum_{i=m+1}^{m+l}\left(i - M\left(t_i; \hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}\right)\right)^2}{l}, \tag{16}$$

and

$$\mathrm{PMSE}\left(\hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}; \boldsymbol{I}\right) = \frac{\sum_{i=m+1}^{m+l}\left(n_i - M\left(t_i; \hat{\boldsymbol{\theta}} \text{ or } \hat{\boldsymbol{\alpha}}\right)\right)^2}{l}, \tag{17}$$

for the time-domain and group data, respectively, where m or n_m software faults were 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. Similarly 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 dataset 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 time periods. Then, we calculated the PMSE for the type-I and type-II NHPP-based SRMs. It was immediately evident that a larger observation point corresponded to a shorter prediction length. In Figures 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 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 tended 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 Figures 6–8, we also plot the predictive behavior of the best existing type-I and the best type-II NHPP-based SRMs in group data, TIDS7. It can be seen that the type-I still tended to falsely predict the number of software faults in the early and middle testing phases. More specifically, in Figures 6 and 7, the type-II NHPP-based SRMs showed an increasing trend, the opposite was true for the type-I, whose predictive trend for future phases becomes very flat. However, in Figure 8, the type-I and type-II showed more similar predictive trends. In general, prediction of unknown trend changes over longer periods of time in the future is essentially difficult for both the type-I NHPP and the type-II NHPP. In contrast, prediction of trend changes over a short period of time is relatively easy, but absolute accuracy cannot be guaranteed.

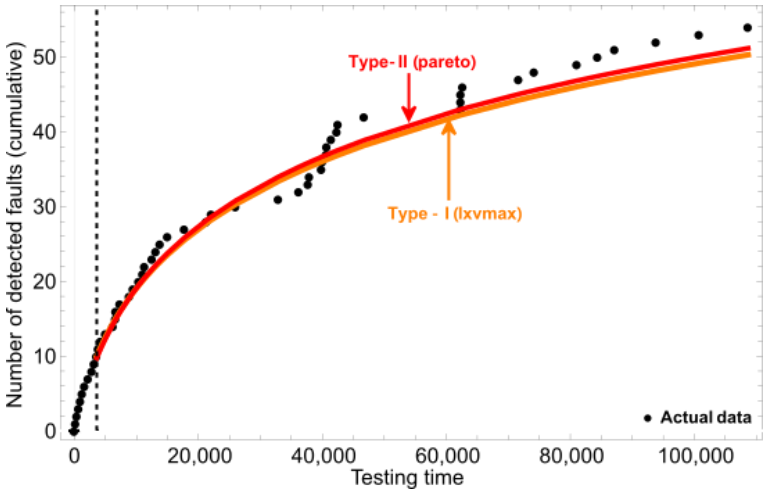

Figure 3. Behavior of fault prediction in TDDS1 (20% observation point).

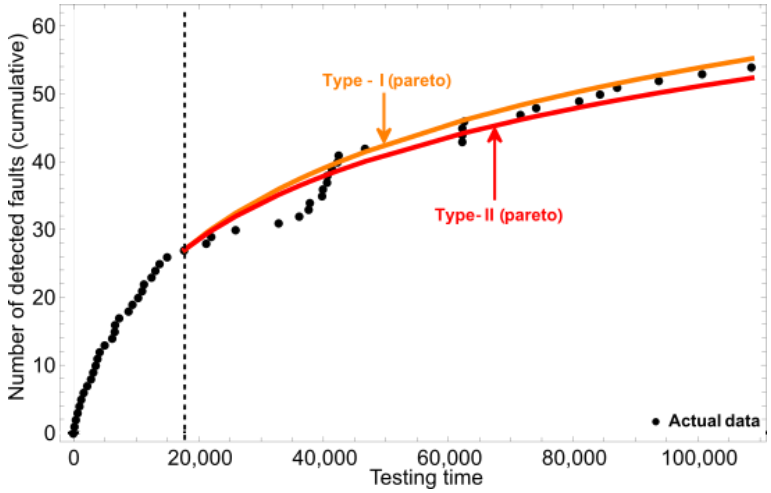

Figure 4. Behavior of fault prediction in TDDS1 (50% observation point).

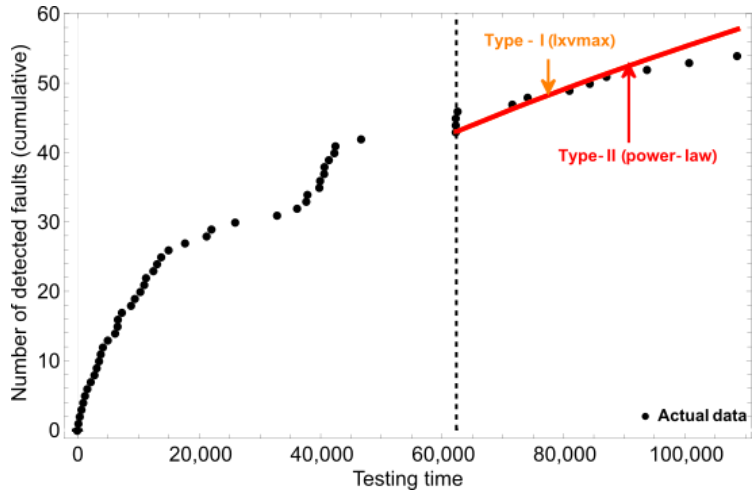

Figure 5. Behavior of fault prediction in TDDS1 (80% observation point).

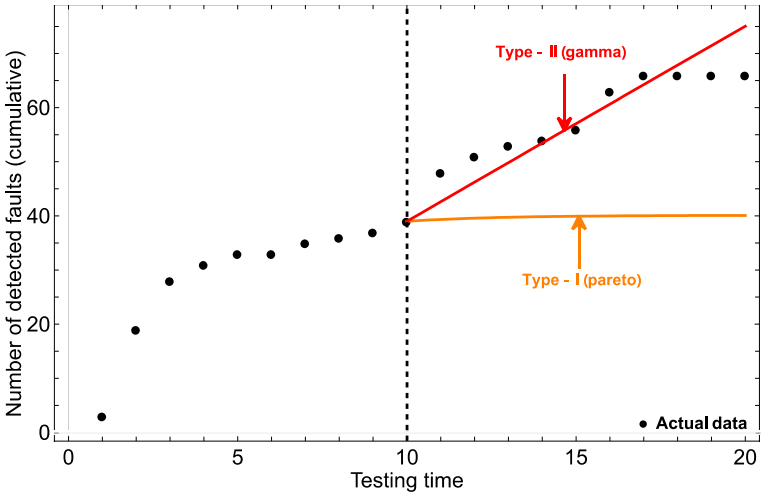

Figure 6. Behavior of fault prediction in TIDS7 (20% observation point).

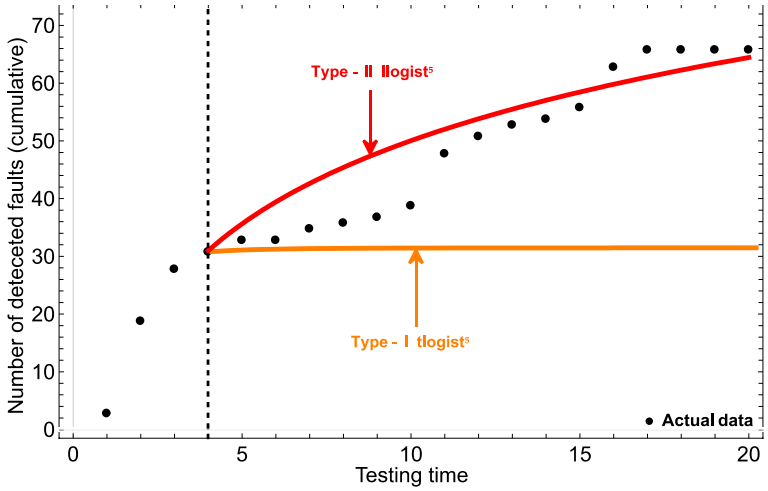

Figure 7. Behavior of fault prediction in TIDS7 (50% observation point).

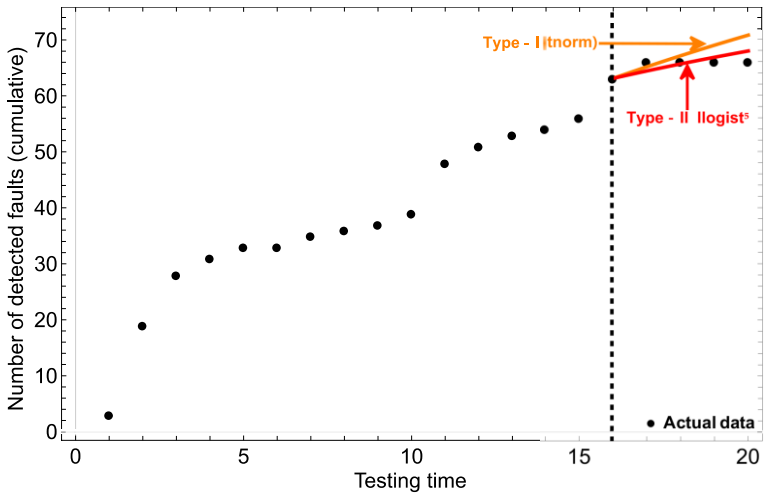

Figure 8. Behavior of fault prediction in TIDS7 (80% observation point).

In Table 7, we present the PMSEs of the best type-I SRM compared to the best type-II SRM in each set of time-domain data. We compared the PMSEs in 11 type-I SRMs and 11 type-II SRMs by selecting the models with the smaller PMSEs as the best SRMs at each observation point. It can be seen that at the 20% observation point, our type-II SRMs provided smaller PMSEs than the existing type-I SRMs in three cases (TDDS2, TDDS6, and TDDS7). During the middle testing phase (at the 50% observation point), we observed that our type-II SRMs outperformed the type-I SRMs in four datasets (TDDS4, TDDS6~TDDS8). As the test proceeded to the late phases (at the 80% observation point), type-II SRMs were able to guarantee smaller PMSEs in TDDS1, TDDS4, and TDDS7. On the other hand, it was found that the best type-II SRMs with better predictive performance than the type-I were all provided by logistic distribution, Musa-Okumoto SRM, and power-law SRM. Upon comparing PMSEs in time-domain data, we believe that the type-II SRMs could become a good alternative to the type-I SRMs.

Table 7. Prediction results in time-domain data.

	20% Observation Point			
	Type-I NHPP		**Type-II NHPP**	
	Best SRM	**PMSE**	**Best SRM**	**PMSE**
TDDS1	Lxvmax	5.073	Musa-Okumoto	6.420
TDDS2	Txvmin	83.964	Llogist	79.614
TDDS3	Tnorm	42.104	Musa-Okumoto	145.648
TDDS4	Lxvmax	32.217	Llogist	207.592
TDDS5	Lnorm	56.477	Musa-Okumoto	198.490
TDDS6	Exp	9177.670	Tlogist	467.320
TDDS7	Lxvmax	1852.520	Lnorm	1474.020
TDDS8	Lxvmax	32.131	Power-law	1417.110
	50% Observation Point			
	Type-I NHPP		**Type-II NHPP**	
	Best SRM	**PMSE**	**Best SRM**	**PMSE**
TDDS1	Pareto	6.118	Musa-Okumoto	6.420
TDDS2	Lxvmax	10.493	Llogist	30.944
TDDS3	Txvmin	5.874	Llogist	11.747
TDDS4	Exp	4480.620	Llogist	18.425
TDDS5	Tlogist	103.504	Cox-Lewis	106.282
TDDS6	Llogist	193.903	Tlogist	77.498
TDDS7	Txvmin	3569.230	Musa-Okumoto	45.344
TDDS8	Pareto	11.712	Musa-Okumoto	10.283
	80% Observation Point			
	Type-I NHPP		**Type-II NHPP**	
	Best SRM	**PMSE**	**Best SRM**	**PMSE**
TDDS1	Lxvmax	5.772	Power-law	3.432
TDDS2	Lxvmax	2.041	Lxvmax	3.697
TDDS3	Lxvmax	0.588	Musa-Okumoto	0.819
TDDS4	Txvmin	6.875	Power-law	4.291
TDDS5	Txvmin	4.253	Cox-Lewis	4.258
TDDS6	Lxvmax	21.715	Power-law	51.677
TDDS7	Lxvmax	57.901	Power-law	9.268
TDDS8	Lxvmax	9.419	Power-law	819.992

As shown in Table 8, when testing in the early phase (at the 20% observation point), it was immediately noticed that our type-II SRMs showed smaller PMSEs than the type-I SRMs in seven out of eight group datasets (except in TIDS3). In addition to logistic-based SRM, Musa-Okumoto SRM, and power-law SRM, which were proven to perform better in Table 7, we observed that Cox-Lewis SRM was also appropriate in some cases of group data (TIDS4 and TIDS5) in terms of predictive performance. At the 50% observation point, we found that the type-II SRMs could guarantee better predictive performance than the type-I SRMs in three cases (TIDS3, TIDS6 and TIDS7). In the late testing phase (at the 80% observation point), only in TIDS2 did our Tlogist type-II SRMs give the smallest PMSE in the future prediction phase. In the group data, the predictive performance of the type-II

SRMs decreased as the software testing proceeded. Hence, it is possible to conclude that the infinite-failure NHPP-based SRMs outperformed the existing finite-failure NHPP-based SRMs for software fault-detection prediction in the early testing phase when group data were available.

Table 8. Prediction results in group data.

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

3.4. 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 NHPP software reliability modeling, the reliability of software at a given time point t_r can be calculated by $R(t_r;\ \boldsymbol{\theta}$ or $\boldsymbol{\alpha})$;

that is, the probability that the software will be failure-free during time interval $t_r - t_m$, which can be written as

$$\begin{aligned} R(t_r; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) &= \Pr\{N(t_r) - N(t_m) = 0\} \\ &= \exp(-[M(t+x; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha}) - M(t; \boldsymbol{\theta} \text{ or } \boldsymbol{\alpha})]), \end{aligned} \tag{18}$$

where for time data and group data, t_m is defined as the time point of the last fault detected during the software test and the calendar time when the test was stopped, respectively. m is the total number of detected faults before the time point t_m. In this numerical experiment, we assumed 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 (18), respectively.

In Tables 9 and 10, 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, respectively. We selected the type-I SRM and the type-II SRM with the smaller AIC in each time domain and group dataset as the best SRMs. We can see that in almost all datasets (except in TDDS1 and TIDS7), the type-I SRMs tended to predict software reliability than our type-II SRMs. In other words, during the time period $t_r - t_m$, the probability of software failure predicted by the type-II NHPP was much higher than the type-I NHPP. This observation indicates that our type-II SRMs tended 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 datasets.

Table 9. Software reliability assessment in time-domain data.

	Type-I NHPP		**Type-II NHPP**	
	Best SRM	**Reliability**	**Best SRM**	**Reliability**
TDDS1	Lxvmax	2.631×10^{-6}	Musa-Okumoto	2.674×10^{-6}
TDDS2	Lxvmax	3.283×10^{-4}	Cox-Lewis	4.694×10^{-8}
TDDS3	Lxvmax	3.687×10^{-3}	Musa-Okumoto	3.751×10^{-7}
TDDS4	Lxvmax	2.453×10^{-4}	Musa-Okumoto	2.398×10^{-4}
TDDS5	Txvmin	4.573×10^{-1}	Cox-Lewis	3.231×10^{-3}
TDDS6	Exp	1.035×10^{-5}	Power-law	2.596×10^{-8}
TDDS7	Pareto	8.971×10^{-6}	Musa-Okumoto	7.736×10^{-6}
TDDS8	Lxvmin	4.592×10^{-5}	Musa-Okumoto	2.516×10^{-10}

Table 10. Software reliability assessment in group data.

	Type-I NHPP		**Type-II NHPP**	
	Best SRM	**Reliability**	**Best SRM**	**Reliability**
TIDS1	Llogist	4.152×10^{-3}	Tlogist	2.217×10^{-25}
TIDS2	Exp	9.832×10^{-4}	Exp	1.234×10^{-4}
TIDS3	Lxvmax	7.236×10^{-5}	Llogist	6.264×10^{-5}
TIDS4	Llogist	6.373×10^{-1}	Llogist	4.052×10^{-10}
TIDS5	Txvmin	9.633×10^{-1}	Tlogist	1.280×10^{-27}
TIDS6	Tlogist	2.816×10^{-1}	Cox-Lewis	3.221×10^{-27}
TIDS7	Lxvmax	1.939×10^{-7}	Llogist	3.892×10^{-7}
TIDS8	Tnorm	3.865×10^{-2}	Cox-Lewis	2.203×10^{-23}

4. Conclusions

Under the infinite-failure assumption, in addition to the well-known Musa-Okumoto model, Cox-Lewis model, and power-law model, in this work we proposed another eight type-II NHPP-based SRMs with eight different software fault-detection time CDFs. By analyzing eight software fault-count time-domain datasets and eight software fault-count time-interval datasets (group data), we investigated the goodness-of-fit performance and predictive performance of our SRMs. We 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 paper had the same software fault-detection CDFs, which has never been addressed in the past literature.

The experimental results confirmed that our type-II NHPP-based SRMs showed better goodness-of-fit performance in some cases. On the other hand, for the group data, the type-II NHPP-based SRMs exhibited rather better predictive ability than the existing type-I NHPP-based SRMs in the early testing phase. However, as software testing progressed, it was found 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 are a good complement to the type-I NHPP-based SRMs for describing the fault-detection process of software systems, while at the same time, they have greater potential in the early software testing phase. On the other hand, we also confirmed that the type-II NHPP tended to make more conservative predictions than the type-I NHPP in software reliability assessment.

We do not believe that there are any significant limitations to the validity in this work. The datasets used for the numerical experiments were collected during the software/system testing phase under careful supervision and with specific objectives. They have been proven to be of high quality [18,23–26]. Both the finite-failure software reliability modeling assumptions and the infinite-failure software reliability modeling assumptions have been shown to be well-founded. The experimental results exhibited by type-I NHPP-based SRMs and type-II NHPP-based SRMs, such as software reliability, also match the actual situation. There is no evidence so far that our proposed SRMs are inapplicable in any type of software.

In the future, we will introduce virtual testing time in the type-II NHPP-based SRMs, which will be beneficial as we continue to explore the potential of the type-II modeling hypothesis.

Author Contributions: Conceptualization, S.L., T.D. and H.O.; methodology, S.L., T.D. and H.O.; validation, S.L., T.D. and H.O.; writing—original draft preparation, S.L.; writing—review and editing, T.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was supported by JST SPRING, Grant Number JPMJSP2132.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kuo, L.; Yang, T.Y. Bayesian computation for nonhomogeneous Poisson processes in software reliability. *J. Am. Stat. Associ.* **1996**, *91*, 763–773. [CrossRef]
2. Goel, A.L.; Okumoto, K. Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures. *IEEE Trans. Reliab.* **1979**, *R-28*, 206–211. [CrossRef]
3. Okamura, H.; Dohi, T.; Osaki, S. Software reliability growth models with normal failure time distributions. *Reliab. Eng. Syst. Saf.* **2013**, *116*, 135–141. [CrossRef]
4. Achcar, J.A.; Dey, D.K.; Niverthi, M. A Bayesian approach using nonhomogeneous Poisson processes for software reliability models. In *Frontiers in Reliability*; World Scientific: Hackensack, NJ, USA, 1998; pp. 1–18. [CrossRef]

5. Ohba, M. Inflection S-Shaped Software Reliability Growth Model. In *Stochastic Models in Reliability Theory*; Springer: Berlin/Heidelberg, Germany, 1984; pp. 144–162. [CrossRef]
6. Gokhale, S.S.; Trivedi, K.S. Log-logistic software reliability growth model. In Proceedings of the Third IEEE International High-Assurance Systems Engineering Symposium (Cat. No. 98EX231), Washington, DC, USA, 13–14 November 1998; pp. 34–41.
7. Goel, A.L. Software reliability models: Assumptions, limitations, and applicability. *IEEE Trans. Softw. Eng.* **1985**, *SE-11*, 1411–1423. [CrossRef]
8. Ohishi, K.; Okamura, H.; Dohi, T. Gompertz software reliability model: Estimation algorithm and empirical validation. *J. Syst. Softw.* **2009**, *82*, 535–543. [CrossRef]
9. Yamada, S.; Ohba, M.; Osaki, S. S-Shaped Reliability Growth Modeling for Software Error Detection. *IEEE Trans. Reliab.* **1983**, *R-32*, 475–484. [CrossRef]
10. Zhao, M.; Xie, M. On maximum likelihood estimation for a general non-homogeneous Poisson process. *Scand. J. Stat.* **1996**, *23*, 597–607.
11. Abdel-Ghaly, A.A.; Chan, P.Y.; Littlewood, B. Evaluation of competing software reliability predictions. *IEEE Trans. Softw. Eng.* **1986**, *SE-12*, 950–967. [CrossRef]
12. Crétois, E.; Gaudoin, O. New Results on Goodness-of-Fit Tests for the Power-Law Process and Application to Software Reliability. *Int. J. Reliab. Qual. Saf. Eng.* **1998**, *5*, 249–267. [CrossRef]
13. Duane, J.T. Learning Curve Approach to Reliability Monitoring. *IEEE Trans. Aerosp.* **1964**, *2*, 563–566. [CrossRef]
14. Littlewood, B. Rationale for a modified Duane model. *IEEE Trans. Reliab.* **1984**, *R-33*, 157–159. [CrossRef]
15. Musa, J.D.; Okumoto, K. A logarithmic Poisson execution time model for software reliability measurement. In Proceedings of the 7th International Conference on Software Engineering, Orlando, FL, USA, 26–29 March 1984; pp. 230–238.
16. Musa, J.D.; Iannino, A.; Okumoto, K. *Software Reliability, Measurement, Prediction, Application*; McGraw-Hill: New York, NY, USA, 1987.
17. Okamura, H.; Dohi, T. SRATS: Software reliability assessment tool on spreadsheet (Experience report). In Proceedings of the 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), Pasadena, CA, USA, 4–7 November 2013; pp. 100–107.
18. Lyu, M.R. *Handbook of Software Reliability Engineering*; IEEE Computer Society Press: Los Alamitos, CA, USA, 1996; Volume 222.
19. Min, X. *Software Reliability Modeling*; World Scientific: Singapore, 1991.
20. Jun, H.; Shigeru, Y.; Shunji, O. Reliability assessment measures based on software reliability growth model with normalized method. *J. Inf. Process.* **1991**, *14*, 178–183.
21. Barlow, R.E.; Proschan, F. *Mathematical Theory of Reliability, 1965*; SIAM: Philadelphia, PA, USA, 1996.
22. Cox, D.; Lewis, P.A.W. *The Statistical Analysis of Series of Events*; Springer: Dordrecht, The Netherlands, 1966.
23. Musa, J.D. *Software Reliability Data*; Technical Report in Rome Air Development Center; 1979.
24. Vouk, M.A. Using reliability models during testing with non-operational profiles. In Proceedings of the 2nd Bellcore/Purdue Workshop on Issues in Software Reliability Estimation; IEEE: Manhattan, NY, USA, 1992; pp. 103–111.
25. Okamura, H.; Etani, Y.; Dohi, T. Quantifying the effectiveness of testing efforts on software fault detection with a logit software reliability growth model. In Proceedings of the 2011 Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement, Nara, Japan, 3–4 November 2011; pp. 62–68.
26. Wood, A. Predicting software reliability. *Computer* **1996**, *29*, 69–77. [CrossRef]