A Comprehensive Analysis of Proportional Intensity-based Software Reliability Models with Covariates (New Developments on Mathematical Decision Making Under Uncertainty)

AUTHOR(S):

Li, Siqiao; Dohi, Tadashi; Okamura, Hiroyuki

A Comprehensive Analysis of Proportional Intensity-based Software Reliability Models with Covariates

Siqiao Li, Tadashi Dohi, Hiroyuki Okamura
Graduate School of Advanced Science and Engineering,
Hiroshima University
Higashi-Hiroshima 739-8527, Japan
{rel-siqiao, dohi, okamu}@hiroshima-u.ac.jp

Abstract

The black-box approach based on stochastic software reliability models is a simple methodology with only software fault data in order to describe the temporal behavior of fault-detection processes, but fails to incorporate some significant development metrics data observed in the development process. In this paper we develop proportional intensity-based software reliability models with time-dependent metrics, and propose a statistical framework to assess the software reliability with the time-dependent covariate as well as the software fault data. The resulting models are similar to the usual proportional hazard model, but possess somewhat different covariate structure from the existing one. We compare these metrics-based software reliability models with eleven well-known non-homogeneous Poisson process models, which are the special cases of our models, and evaluate quantitatively the goodness-of-fit and prediction. As an important result, the accuracy on reliability assessment strongly depends on the kind of software metrics used for analysis and can be improved by incorporating the time-dependent metrics data in modeling.

1. INTRODUCTION

During the last three decades, the stochastic models, called *software reliability models* (SRMs) that analyze and explain the software fault-detection phenomena, have been extensively developed in the literature [24], [28], [35]. In fact, till now, over 200 SRMs have been proposed from various mathematical standpoints. The classical and the most important SRMs may be the non-homogeneous Poisson process (NHPP) models that have gained popularity for describing the stochastic behavior of the number of software faults detected in the testing phase. The representative NHPP-based SRMs are characterized by the mean value functions, which are proportional to the cumulative distribution functions (CDF) of software fault-detection time. Since the seminal contribution by Goel and Okumoto [11], many authors proposed NHPP-based SRMs under different model assumptions. The representative NHPP-based SRMs assumed the exponential CDF [11], the gamma CDF [43, 44], the truncated-logistic CDF [30], the log-logistic CDF [12], the Pareto CDF [1], the truncated-normal CDF [33], the log-normal CDF [2, 33], the extreme-value CDFs [31] including the Weibull CDF [10]. These NHPP SRMs are based on different debugging scenarios from each other, and can catch qualitatively typical (but not general) reliability growth phenomena observed in the testing phases of software products. In other words, the black-box approach based on the NHPP-based SRMs is a simple methodology with only software fault data in order to describe the temporal behavior of fault-detection processes, but fails to incorporate some significant development metrics data observed in the development process.

On the other hand, much effort has been spent to clarify the relationship between the software quality and some kinds of software metrics. McCabe [25] proposes several software engineering metrics which are units of measurement to characterize software products, processes and human resources in the development. Halstead [13] points out the significance of software science and derives deterministic equations to estimate the number of residual faults in software with programming effort. Putnam [37] and Takahashi and Kamayachi [42] show empirical relationships between the software fault characteristics and the so-called environmental factors to characterize the software products, such as programmer skill, programming language, coding techniques, reusability of existing code, *etc*. Pillai and Nair [36] discuss an estimation problem of software cost and development effort with several metrics data.

However, the approaches mentioned above are essentially the deterministic ones and cannot represent uncertainty of software fault-detection processes in testing. In general, the software metrics can be classified into four categories: product metrics, development metrics, deployment & usage metrics and software-hardware configurations metrics, so that the utilization of these information as well as the fault-detection data will lead to the accurate reliability assessment of software products in practice.

The stochastic approach to incorporate the software development metrics and/or environmental factors is taken by Ascher [4], [5], Bendell [6], Evanco and Lacovara [9], Evanco [8] and Nishio and Dohi [29]. They utilize the proportional hazard model (PHM) or equivalently Cox regression model [7], and formulate the software fault-detection time distribution by regarding the time-series metrics data as the covariate [21], [26]. Pham [35] develops an enhanced proportional hazard SRM based on a continuous-time Markov chain and considers a dynamic version of PHM. However, if the cumulative effect of software development/test effort reported in [27] is considered for analysis, the above modeling approach based on the PHMs loses their validation because the covariate to represent the development effort usually consists of 0-1 binary values. Shibata, Rinsaka and Dohi [41] introduced the discrete proportional hazard model on a cumulative Bernoulli trial process, and to represent a generalized fault-detection processes having time-dependent covariate structure. Okamura *et al.* [34] proposed a discret-time multi-factor SRM based on logistic regression and its effective statistical parameter estimation method. Kuwa and Dohi [19, 20] extended the existing logit regression-based SRM and Cox proportional hazards regression-based SRM to help them improve goodness-of-fit and predictive performances. Khoshgoftaar and Munson [16] and Khoshgoftaar *et al.* [14], [17], [18] develop the linear and non-linear regression models to quantify the software quality with some metrics data. Li *et al.* [23] discuss a metrics-based modeling method to predict the field defect-occurrence rate by using the classical moving average and the exponential smoothing. Khoshgoftaar *et al.* [15] apply the Poisson regression model and the zero-inflated Poisson regression model to predict the rank-order of software modules. Schnieidewind [39], [40] also use the regression models to evaluate the software maintenance process with measurement. Amasaki *et al.* [3] classify the trend of fault data with the rank correlation coefficient and apply the logistic regression model to assess the software quality, which is measured by the number of detected faults after shipment. In this way, various regression approaches have been used to develop the metrics-based SRMs.

Rinsaka *et al.* [38] developed a proportional intensity-based SRM with both software fault data and testing metrics data. The proportional intensity model (PIM) is proposed by Lawless [22] as a natural extension of the usual PHM, where the proportional hazard function in the Cox regression is used for the intensity function of the NHPP. The most different point from the existing PHM is that PIM is a dynamic SRM to describe the software reliability growth phenomenon. That is, they introduce the Cox regression to incorporate the time-dependent development/test metrics instead of the linear and non-linear regression with white noise or the logistic regression, and still utilize the stochastic counting process to describe the cumulative number of faults detected in the software testing (note that the regression models in [3, 14–18, 39, 40] are static models in time). In that sense, PIM proposed here would possess both applicability to the actual software reliability assessment from the similarity to the NHPP-based SRMs and flexibility to incorporate the time-dependent metrics data observed in the testing phase.

In this paper, we develop 11 proportional intensity NHPP-based SRMs with time-dependent software metrics, which could incorporate multiple time-dependent cumulative/non-cumulative software development metrics data. The baseline intensity functions of our SRMs are consistent with the 11 existing NHPP-based SRMs [1, 2, 11, 12, 30, 31, 33, 43, 44]. The rest part of this paper proceeds as follows. In Section 2, we describe the basic NHPP-based SRMs and refer to the statistical estimation methods of model parameters. In Section 3 we introduce PIM and give an additive cumulative intensity structure to model the cumulative effect of software development/test effect. Section 4 is devoted to present numerical examples with real software fault data and the time-dependent test metrics, where test execution time (CPU hr), failure identification work (person hr), computer time-failure identification (CPU hr) are used as test metrics to compare our proportional intensity-based SRMs with the existing NHPP-based SRMs. We estimate the model parameters by utilizing the maximum likelihood estimation and evaluate the performance metrics in terms of goodness-of-fit and prediction. As an important result, it is shown empirically that the accuracy on reliability assessment strongly depends on the kind of software metrics used for analysis and can be improved by incorporating the time-dependent metrics data in modeling. In fact, it is concluded that our proportional intensity NHPP-based SRMs outperform the existing NHPP-based SRMs in terms of information criteria and prediction ability.

2. NHPP-Based Software Reliability Model

Let $\{N(t), t \geq 0\}$ denote the number of software faults detected by time t in the software testing and be a stochastic counting process satisfying:

(i) $N(0) = 0$

(ii) $\{N(t), t \geq 0\}$ has independent increment

(iii) $\Pr\{N(t + \Delta t) - N(t) \geq 2\} = o(\Delta t)$

(iv) $\Pr\{N(t + \Delta t) - N(t) = 1\} = \lambda(t)\Delta t + o(\Delta t)$,

where $\lambda(t)$ is an absolutely continuous (deterministic) function of only time t, and $o(\Delta t)$ is the higher order term of infinitesimal time Δt, so that

$$\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0. \tag{1}$$

Then the counting process is said the non-homogeneous Poisson process (NHPP) with intensity function $\lambda(t)$. Since this is a typical Markov process with time-dependent transition rate, it is easily derived that

$$\Pr\{N(t) = n\} = \frac{\{H(t)\}^n}{n!} \exp\{-H(t)\}, \tag{2}$$

where

$$H(t) = \mathrm{E}[N(t)] = \int_0^t \lambda(x)dx, \quad H(0) = 0 \tag{3}$$

is said the mean value function and means the expected cumulative number of faults detected by time t.

It is assumed that each software fault is detected at independent and identically distributed (i.i.d.) random time with a non-degenerate cumulative distribution function (CDF), $F(t; \boldsymbol{\alpha})$, having the parameter $\boldsymbol{\alpha}$, and that the residual number of software faults at time $t = 0$ is a Poisson distributed random variable with parameter ω (> 0). Then the resulting software fault detection process obeys the NHPP with mean value function $H(t; \boldsymbol{\theta}) = \omega F(t; \boldsymbol{\alpha})$ with $\boldsymbol{\theta} = (\omega, \boldsymbol{\alpha})$. In this way, the commonly used assumption in software reliability engineering is that the initial number of residual software faults in a software system is expected to be finite, *i.e.*, $\lim_{t\to\infty} H(t; \boldsymbol{\theta}) = \omega$ (> 0). In the classical software reliability modeling, the main research issue was to determine the intensity function $\lambda(t; \boldsymbol{\theta})$, or equivalently the mean value function $H(t; \boldsymbol{\theta})$ so as to fit the software fault data. and Dohi [32] implemented the existing NHPP-based SRMs with 11 software fault-detection time CDFs in the software reliability assessment tool on the spreadsheet (SRATS), which includes exponential (exp), gamma, Pareto, log-normal (lnorm), log-logistic (llogist), log-extreme-value minimum (lxvmin), log-extreme-value maximum (lxvmax), truncated logistic (tlogist), truncated normal (tnorm), truncated extreme-value minimum (txvmin), truncated extreme-value maximum (txvmax) distributions. In Table 1, we summarize these 11 NHPP-based SRMs with their baseline intensity functions.

Table 1: The existing NHPP-based SRMs.

Models	$\lambda(t; \boldsymbol{\theta})$ & $F(t; \boldsymbol{\alpha})$
Exponential dist. (exp) [11]	$\lambda(t; \boldsymbol{\theta}) = \omega b e^{-bt}$ $F(t; \boldsymbol{\alpha}) = 1 - e^{-bt}$
Gamma dist. (gamma) [43], [44]	$\lambda(t; \boldsymbol{\theta}) = \omega \frac{e^{-\frac{t}{c}}(\frac{t}{c})^{b-1}}{c\Gamma(b)}$ $F(t; \boldsymbol{\alpha}) = \int_0^t \frac{c^b s^{b-1} e^{-cs}}{\Gamma(b)} ds$
Pareto dist. (pareto) [1]	$\lambda(t; \boldsymbol{\theta}) = \frac{\omega bc\left(\frac{c}{c+t}\right)^{b-1}}{(c+t)^2}$ $F(t; \boldsymbol{\alpha}) = 1 - (\frac{b}{t+b})^c$
Truncated normal dist. (tnorm) [33]	$\lambda(t; \boldsymbol{\theta}) = \frac{\omega e^{-\frac{(c-t)^2}{2b^2}}}{\sqrt{2\pi}b\left(1-\frac{1}{2}\mathrm{erfc}\left(\frac{-c}{\sqrt{2}b}\right)\right)}$ $F(t; \boldsymbol{\alpha}) = \frac{1}{\sqrt{2\pi}b} \int_{-\infty}^t e^{-\frac{(s-c)^2}{2b^2}} ds$
Log-normal dist. (lnorm) [2], [33]	$\lambda(t; \boldsymbol{\theta}) = \frac{\omega e^{-\frac{(c-\log(t))^2}{2b^2}}}{\sqrt{2\pi}bt}$ $F(t; \boldsymbol{\alpha}) = \frac{1}{\sqrt{2\pi}b} \int_{-\infty}^t e^{-\frac{(s-c)^2}{2b^2}} ds$
Truncated logistic dist. (tlogist) [30]	$\lambda(t; \boldsymbol{\theta}) = \frac{\omega e^{-\frac{t-c}{b}}}{b\left(1-\frac{1}{e^{c/b}+1}\right)\left(e^{-\frac{t-c}{b}}+1\right)^2}$ $F(t; \boldsymbol{\alpha}) = \frac{1-e^{-bt}}{1+ce^{-bt}}$
Log-logistic dist. (llogist) [12]	$\lambda(t; \boldsymbol{\theta}) = \frac{\omega e^{-\frac{\log(t)-c}{b}}}{bt\left(e^{-\frac{\log(t)-c}{b}}+1\right)^2}$ $F(t; \boldsymbol{\alpha}) = \frac{(bt)^c}{1+(bt)^c}$
Truncated extreme-value max dist. (txvmax) [31]	$\lambda(t; \boldsymbol{\theta}) = \frac{\omega e^{\frac{t-c}{b}-e^{-\frac{t-c}{b}}}}{b\left(1-e^{-e^{c/b}}\right)}$ $F(t; \boldsymbol{\alpha}) = e^{-e^{-\frac{t-c}{b}}}$
Log-extreme-value max dist. (lxvmax) [31]	$\lambda(t; \boldsymbol{\theta}) = \frac{\omega c e^{-\left(\frac{t}{b}\right)^{-c}}\left(\frac{t}{b}\right)^{-c-1}}{b}$ $F(t; \boldsymbol{\alpha}) = e^{-\left(\frac{t}{b}\right)^{-c}}$
Truncated extreme-value min dist. (txvmin) [31]	$\lambda(t; \boldsymbol{\theta}) = \frac{\omega e^{-\frac{c-t}{b}-e^{-\frac{c-t}{b}}+e^{c/b}}}{b}$ $F(t; \boldsymbol{\alpha}) = e^{-e^{-\frac{t-c}{b}}}$
Log-extreme-value min dist. (lxvmin) [10]	$\lambda(t; \boldsymbol{\theta}) = \frac{\omega e^{-\frac{c-\log(t)}{b}-e^{-\frac{c-\log(t)}{b}}}}{bt}$ $F(t; \boldsymbol{\alpha}) = e^{-e^{-\frac{bt}{b}-\frac{t-c}{b}}}$
	$(\omega{>}0, b{>}0, c{>}0)$

Let y_k $(k = 1, 2, \cdots, n)$ be the total number of software faults detected by each testing time t_k measured by calendar time. For the parameter set $\boldsymbol{\theta} = (a, b)$, the mean value function of the NHPP is represented by $H(t; \boldsymbol{\theta})$. The commonly used technique to estimate the model parameter $\boldsymbol{\theta}$ is the maximum likelihood method. For convenience, let us define $(t_0, y_0) = (0, 0)$ without any loss of generality. Since the likelihood function is given by

$$\begin{aligned} L(\boldsymbol{\theta}) &= \Pr\{N(t_1) = y_1, \cdots, N(t_n) = y_n\} \\ &= \exp[-H(t_n; \boldsymbol{\theta})] \prod_{k=1}^{n} \\ &\quad \times \frac{\{H(t_k; \boldsymbol{\theta}) - H(t_{k-1}; \boldsymbol{\theta})\}^{y_k - y_{k-1}}}{(y_k - y_{k-1})!}, \end{aligned} \tag{4}$$

taking the logarithm of both sides of Eq.(4) yields the logarithmic likelihood function:

$$\begin{aligned} \mathrm{LLF}(\boldsymbol{\theta}) = \sum_{k=1}^{n} (y_k - y_{k-1}) \ln[H(t_k; \boldsymbol{\theta}) \\ - H_t(t_{k-1}; \boldsymbol{\theta})] - H(t_n; \boldsymbol{\theta}) \end{aligned}$$

$$- \sum_{k=1}^{n} \ln[(y_k - y_{k-1})!]. \quad (5)$$

Hence the ML estimate $\hat{\boldsymbol{\theta}}$ is given by the solution of $\mathrm{argmax}_{\boldsymbol{\theta}} \mathrm{lnLLF}(\boldsymbol{\theta})$.

3. Proportional Intensity Model

3.1. Model Description

Here we develop a novel SRM to incorporate the multiple testing-effort parameters, which is consistent to the maximum likelihood estimation. Suppose that l (≥ 1) kinds of software metrics data $\mathbf{x}_k = (x_{k1}, \cdots, x_{kl})$ ($k = 1, 2, \cdots$) are available at each testing time t_k ($= 0, 1, 2, \cdots$). It is also assumed that each metrics $\mathbf{x}_k$ depends on the cumulative testing time t_k and can be regarded as a function of time, say, $\mathbf{x}_k(t_k)$. In statistics, this type of parameter is called the time-dependent covariate [21], [26] and has been studied extensively in the context of Cox PHM. For the NHPP in Eq.(2), we assume the following intensity function:

$$\lambda_x(t_k; \boldsymbol{\theta}, \boldsymbol{\beta} \mid \mathbf{x}_k) = \lambda_0(t_k; \boldsymbol{\theta}) g(\mathbf{x}_k; \boldsymbol{\beta}), \quad (6)$$

where $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_l)$. In Eq.(6), the function $\lambda_0(t_k; \boldsymbol{\theta})$ (> 0) is called the baseline intensity function and is a function of only time. On the other hand, the function $g(\mathbf{x}_k; \boldsymbol{\beta})$ (> 0) is called the covariate function and is a function of the software time-series metrics $\mathbf{x}_k(t_k)$ and the coefficient parameter $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_l)^T$. Similar to the usual Cox's PHM, an appropriate choice of the covariate function would be given by the following exponential form:

$$g(\mathbf{x}_k; \boldsymbol{\beta}) = \exp(\mathbf{x}_k \boldsymbol{\beta}). \quad (7)$$

Actually this form is well known to be convenient for analysis and to be rather flexible to express the covariate structure in many application [7], [21], [26]. Lawless [22] also assumes the above exponential covariate structure and analyzes the real statistical data in medical applications.

However, it is worth noting that the time-independent covariate considered by Lawless [22] is also the 0-1 binary value and does not deal with the cumulative value like test execution time (CPU hr), *etc*. In other words, a new modeling framework is needed for analysis of time-series metrics data. The simplest but reasonable model is to take account of an effect of the cumulative number of faults in an expression of the mean value function. Suppose:

$$\begin{aligned} H_p(t_1; \boldsymbol{\theta}, \boldsymbol{\beta}) &= \int_0^{t_1} \lambda_0(u; \boldsymbol{\theta}) \exp(\mathbf{x}_1 \boldsymbol{\beta}) du, \quad (8) \\ H_p(t_2; \boldsymbol{\theta}, \boldsymbol{\beta}) &= H_p(t_1; \boldsymbol{\theta}, \boldsymbol{\beta}) \\ &\quad + \int_{t_1}^{t_2} \lambda_0(u; \boldsymbol{\theta}) \exp(\mathbf{x}_2 \boldsymbol{\beta}) du, \quad (9) \\ &\vdots \\ H_p(t_k; \boldsymbol{\theta}, \boldsymbol{\beta}) &= \sum_{i=1}^{k} \exp(\mathbf{x}_i \boldsymbol{\beta}) \int_{t_{i-1}}^{t_i} \lambda_0(u; \boldsymbol{\theta}) du \\ &= \sum_{i=1}^{k} \exp(\mathbf{x}_i \boldsymbol{\beta}) \\ &\quad \times [H_0(t_i; \boldsymbol{\theta}) - H_0(t_{i-1}; \boldsymbol{\theta})], \quad (10) \end{aligned}$$

where $H_0(t_i; \boldsymbol{\theta}) = \int_0^{t_i} \lambda_0(u; \boldsymbol{\theta}) du$. Note that when $\beta_j = 0$ for all j ($= 1, 2, \cdots, l$), the above PIM can be reduced to the existing NHPP-based SRM.

3.2. Maximum likelihood estimation

Both the model parameters $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ can be estimated by a method of maximum likelihood. Suppose that n set of fault-detection data (t_k, y_k) ($k = 0, 1, 2, \cdots, n$) and $l \times n$ software metrics data $\mathbf{x}_k = (x_{k1}, \cdots, x_{kl})$ can be observed for testing time interval $(0, t_k]$, where, y_k is the cumulative number of detected software faults and $(x_{k1}, \cdots, x_{kl})$ are l kinds of software metrics data consumed until time t_k. Under the above assumptions and the property of independent increment of the NHPP, the likelihood function for PIM with mean value function $H_p(t)$ is given by

$$\begin{aligned} L(\boldsymbol{\theta}, \boldsymbol{\beta}) &= \Pr\{N(t_1) = y_1, \cdots, N(t_n) = y_n\} \\ &= \exp[-H_p(t_n; \boldsymbol{\theta}, \boldsymbol{\beta})] \\ &\quad \times \prod_{k=1}^{n} \frac{1}{(y_k - y_{k-1})!} \\ &\quad \times \{H_p(t_k; \boldsymbol{\theta}, \boldsymbol{\beta}) - H_p(t_{k-1}; \boldsymbol{\theta}, \boldsymbol{\beta})\}^{y_k - y_{k-1}} \end{aligned} \quad (11)$$

where, $(t_0, y_0) = (0, 0)$ and $x_{0j} = 0$ ($j = 1, 2, \cdots, l$). Taking the logarithm of both sides of Eq.(11), we have

$$\begin{aligned} &\mathrm{LLF}(\boldsymbol{\theta}, \boldsymbol{\beta}) \\ &= \sum_{k=1}^{n} (y_k - y_{k-1}) \ln[H_p(t_k; \boldsymbol{\theta}, \boldsymbol{\beta}) \\ &\quad - H_p(t_{k-1}; \boldsymbol{\theta}, \boldsymbol{\beta})] - H_p(t_n; \boldsymbol{\theta}, \boldsymbol{\beta}) \\ &\quad - \sum_{k=1}^{n} \ln[(y_k - y_{k-1})!]. \end{aligned} \quad (12)$$

Since the maximum likelihood estimates $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}})$ can be obtained via the direct maximization of the logarithmic likelihood $\mathrm{LLF}(\boldsymbol{\theta}, \boldsymbol{\beta})$ without applying the least squared sum method, we can enjoy consistently the rich property of the maximum likelihood estimation in the PIM framework.

4. Numerical Examples

In this section, we focus on four real data sets collected in the actual software development projects for the real time command and control systems [27] in Table 2. In these four data sets, we perform goodness-of-fit tests of PIM and evaluate the predictive performances with cumulative/non-cumulative time-dependent metrics data as the covariates, say, $\mathbf{x}_k = (x_{k1}, \cdots, x_{kl})/\mathbf{x}_k = \left(x_{k1} - x_{(k-1)1}, \cdots, x_{kl} - x_{(k-1)l}\right)$. l is the number of time-dependent metrics data in each data set and $k = 0, 1, 2, \cdots, n$.

4.1. Goodness-of-fit Performances

For PIM, we assume 11 kinds of baseline intensity functions in Table 1. To investigate the effect of each time-dependent metric data on the stochastic behavior of the cumulative number of faults detected in the testing phase, we calculate the maximum likelihood estimates $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}})$ of covariate equations $g(\mathbf{x}_k; \boldsymbol{\beta}) = \exp(\mathbf{x}_k \boldsymbol{\beta})$ for every metrics data combinations. Since, each data set presented in Table 2 contains three kinds of time-dependent metric data, so we need to consider a total of 7 combinations as shown in Table 3. By deriving the corresponding log likelihood (LLF), we investigate the goodness-of-fit performance with two measures: Akaike information criterion (AIC) and mean squared error (MSE), where.

$$\mathrm{AIC} = -2\mathrm{LLF}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}}) + 2\pi, \tag{13}$$

and

$$\mathrm{MSE} = \frac{1}{n}\sqrt{\sum_{k=1}^{n}(y_k - \hat{H}_p(t_k; \boldsymbol{\theta}, \boldsymbol{\beta}))^2}. \tag{14}$$

π is the number of free parameters. The smaller AIC/MSE is the better SRM in terms of the goodness-of-fit to the underlying fault count data.

Figure 1 illustrates the estimated mean value functions and the cumulative number of software faults detected in GDS1. The best SRMs with minimum AIC were selected from the our 11 proportional intensity NHPP-based SRMs with cumulative metric data (red curve), 11 proportional intensity NHPP-based SRMs with non-cumulative metric data (blue curve) and the existing NHPP-based SRMs in SRATS (orange curve). At first glance, the three curves exhibit almost similar behavior, but a closer look reveals that our PIMs can represent more complex behaviors than existing NHPP-based SRM. Then, in Figure 2, we also plot the behavior of estimated number of detected fault counts in each testing time interval for the best SRMs in GDS1. The orange bar-chart represents the actual software detected faults data in each time interval. This figure

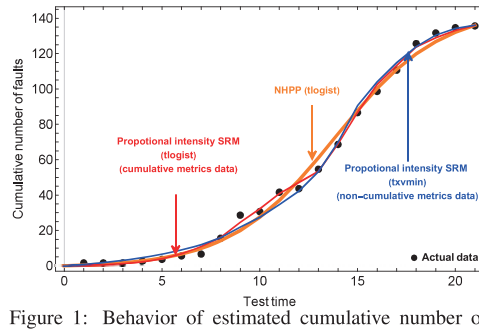

Figure 1: Behavior of estimated cumulative number of software faults in GDS1.

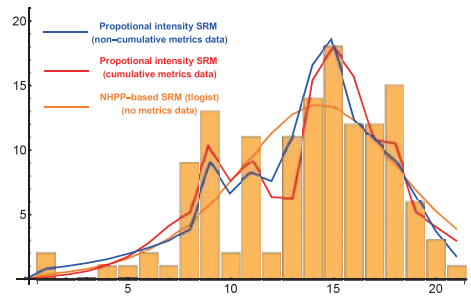

Figure 2: Behavior of estimated number of detected fault counts during time interval in GDS1.

clearly shows that our two proportional intensity NHPP-based SRMs have the more better goodness-of-fit performances then existing NHPP-based SRM.

We present the best AIC results for four time-dependent metrics data in Table 4 for more accurate comparisons. By comparing our two types of proportional intensity NHPP-based SRMs with the existing NHPP-based SRMs in SRATS [32], the bold font is utilized to marks the best SRM with minimum AIC in each data set. It can be seen that in the all data sets, our proportional intensity NHPP-based SRMs could provide the better goodness-of-fit performances than the existing NHPP-based SRMs in SRATS. Especially, the proportional intensity SRMs with non-cumulative metrics data could give the smaller AIC (GDS1, GDS2, and GDS3) and smaller MSE (GDS2, GDS3, and GDS4) in 3 cases. We can also notice that our SRMs with combination II, III and IV covariate equations could guarantee the better goodness-of-fit results in most cases. Therefore, our conclusion is more inclined to think that are our proportional intensity NHPP-based SRMs are attractive in software reliability modeling and should be competitors with the high potential ability for the existing NHPP-based SRMs. The execution time and failure identification work can effectively help us to improve the goodness-of-fit performances of our SRMs.

Table 2: Data sets.

Data	No. faults	Testing days	Metrics data
GDS1	136	21	Execution time (CPU hr), failure identification work (person hr), computer time-failure identification (CPU hr)
GDS2	54	17	Execution time (CPU hr), failure identification work (person hr), computer time-failure identification (CPU hr)
GDS3	38	14	Execution time (CPU hr), failure identification work (person hr), computer time-failure identification (CPU hr)
GDS4	53	16	Execution time (CPU hr), failure identification work (person hr), computer time-failure identification (CPU hr)

Table 3: Combinations of covariate equation $g(\mathbf{x}_{kl};\boldsymbol{\beta})$.

	$g(\mathbf{x}_{kl};\boldsymbol{\beta})(l=1,2,3)$
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.	

4.2. Predictive Performances

Next, we investigate the predictive ability of proportional intensity NHPP-based SRMs. On the observation point n' $(1 \leq n' < n)$ when 50% or 80% of all data are available, we predict the future behavior of the cumulative number of faults. To assess the predictive ability, we apply the prediction squares error (PMSE) as predictive performance measure, where

$$\text{PMSE} = \frac{1}{n-\acute{n}}\sqrt{\sum_{k=\acute{n}+1}^{n}\left[y_k - \hat{H}_p(t_k;\boldsymbol{\theta},\boldsymbol{\beta})\right]^2}. \quad (15)$$

t is obvious that the smaller PMSE indicates the better predictive performance.

In order to predict the future behavior of software fault-detection process, it should be noted that estimates of development/test metrics are required, since the mean value functions of PIM depend on the covariates. In practice, three cases are possible to consider.

Case I: All the test/development metrics data are completely known and fixed in advance, so that the software testing expenditures are given before testing.

Case II: The test/development metrics data do not change in the future.

Case III: The test/development metrics data experienced in future are random variables and can be predicted.

Especially, in the case III, we are requested to introduce additional probability models on test/development metrics data. In this paper, we made the two assumptions and employ the linear and exponential regression methods to predict the future test/development metrics data. Becasue the exponential regression is not suitable for making prediction on non-cumulative data (the variable may be 0, causing the correlation coefficient to not be calculated), so, we totally consider 7 cases of estimated development/test metrics data in future phase under the above three assumptions, to help us investigate the predictive performances of our proportional intensity NHPP-based SRMs with observed cumulative metrics data and non-cumulative metrics data.

Tables 5 and 6 present the comparison results on the PMSE in four data sets at 50 % observation point and 80 % observation point, respectively, where we select the best SRM with the smallest PMSE from the our proportional intensity NHPP-based SRMs and the existing NHPP-based SRMs. Bold font is utilized to marks the best SRMs with minimum PMSE in each data set. From these two tables, it can seen that our proportional intensity NHPP-based SRMs can still outperform the existing NHPP-based SRMs in all the data sets. We can also find that utilizing the estimated metrics data under Case II (the test/development metrics data do not change in the future) assumption in the future phase tends to give the more better predictive performances than the other two assumptions in more than half data sets (GDS1 50%, GDS2 50%, GDS3 50%, GDS4 50% and GDS4 80%). 5 out of 8 (GDS2 50%, GDS4 50%, GDS2 80%, GDS3 80% and GDS4 80%) minimum PMSEs are given by our proportional intensity NHPP-based SRMs with non-cumulative metrics data. The SRMs with Combination II covariate function could provide the best PMSEs in two (GDS1 50% and GDS2 50%) and three (GDS1 80%, GDS3 80% and GDS4 80%) data sets, respectively. The remaining three best PMSEs are given by the SRMs with combinations V, VI and VII covariate functions. Combining Table 4, we can find that failure identification work is the most important development metric data, which can effectively improve the software faults prediction accuracy of our SRM in the future phase.

5. Conclusions

In this paper, we have developed the proportional intensity NHPP-based SRMs with 11 well-known underline

Table 4: Goodness-of-fit performances based on AIC.

(i) Best proportional intensity model (cumulative metrics data)				
	Model	AIC	MSE	$\hat{\boldsymbol{\beta}}$
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	txvmin	63.208	0.553	-
GDS4	txvmin	79.761	0.530	-

intensity functions, which could incorporate multiple time-dependent cumulative/non-cumulative software development/test metrics data. In our numerical experiments, by regarding three types of time-dependent metrics data as the covariates in totally four data sets, we compared our proportional intensity NHPP-based SRMs with the existing SRMs in the past literature in terms of goodness-of-fit and predictive performances. We have confirmed that our proportional intensity NHPP-based SRMs could show the better performances in all the cases than the existing 11 NHPP-based SRMs in SRATS. We also confirmed that failure identification work is the most important development/test metric that can help us improve the accuracy of our SRMs, both in terms of goodness-of-fit and software failure prediction in future phase.

References

[1] A. A. Abdel-Ghaly, P. Y. Chan, and B. Littlewood. Evaluation of competing software reliability predictions. *IEEE Transactions on Software Engineering*, SE-12(9):950–967, 1986.

[2] J. A. Achcar, D. K. Dey, and M. Niverthi. A Bayesian approach using nonhomogeneous Poisson processes for software reliability models. In *Frontiers in Reliability*, pages 1–18. World Scientific, 1998.

[3] S. Amasaki, T. Yoshitomi, O. Mizuno, Y. Takagi, and T. Kikuno. A new challenge for applying time series metrics data to software quality estimation. *Software Quality Journal*, 13(2):177–193, 2005.

[4] H. Ascher. Proportional hazards modelling of software failure data. *Software Reliability; State of the Art Report (A. Bendell and P. Mellor, eds.)*, pages 229–263, 1986.

[5] H. Ascher. The use of regression techniques for matching reliability models to the real world. *Software System Design Methods, NATO ASI Series (J. K. Skwirzynski, ed.)*, F22:366–378, 1986.

[6] A. Bendell. The use of exploratory data analysis techniques for software reliability assessment and prediction. *Software System Design Methods, NATO ASI Series (J. K. Skwirzynski, ed.)*, F22:337–351, 1986.

[7] D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society*, B-34:187–220, 1972.

[8] W. M. Evanco. Using a proportional hazards model to analyze software reliability. In *Proc. 9th Int'l Conf. Software Technology & Engineering Practice*, pages 134–141. IEEE CS Press, 1999.

[9] W. M. Evanco and R. Lacovara. A model-based framework for the integration of software metrics. *Journal of Systems and Software*, 26:75–84, 1995.

[10] A. L. Goel. Software reliability models: assumptions, limitations, and applicability. *IEEE Transactions on Software Engineering*, SE-11(12):1411–1423, 1985.

[11] A. L. Goel and K. Okumoto. Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability*, R-28(3):206–211, 1979.

[12] S. S. Gokhale and K. S. Trivedi. Log-logistic software reliability growth model. In *Proceedings Third IEEE International High-Assurance Systems Engineering Symposium (HASE 1998)*, pages 34–41, 1998.

[13] M. H. Halstead. *Elements of Software Science*. Elsevier, New York, 1977.

[14] T. M. Khoshgoftaar, B. B. Bhattacharyya, and G. D. Richardson. Predicting software errors, during devel-

Table 5: Predictive performances 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 6: Predictive performances 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

opment, using nonlinear regression models: a comparative study. *IEEE Transactions on Reliability*, 41(3):390–395, 1992.

[15] T. M. Khoshgoftaar, K. Gao, and R. Szabo. Comparing software fault predictions of pure and zero-inflated Poisson regression models. *International Journal of Systems Science*, 36(11):705–715, 2005.

[16] T. M. Khoshgoftaar and J. C. Munson. Predicting software development errors using software complexity metrics. *IEEE Journal of Selected Areas in Communications*, 8(2):253–261, 1990.

[17] T. M. Khoshgoftaar, J. C. Munson, B. B. Bhattacharyya, and G. D. Richardson. Predictive modeling techniques of software quality from software measures. *IEEE Transactions on Software Engineering*, 18(11):979–987, 1992.

[18] T. M. Khoshgoftaar, A. Pandya, and D. Lanning. Application of neural networks for predicting program fault. *Annals of Software Engineering.*, 1(1):141–154, 1995.

[19] D. Kuwa and T. Dohi. Generalized logit regression-based software reliability modeling with metrics data. In *2013 IEEE 37th Annual Computer Software and Applications Conference*, pages 246–255, 2013.

[20] D. Kuwa, T. Dohi, and H. Okamura. Generalized cox proportional hazards regression-based software reliability modeling with metrics data. In *2013 IEEE 19th Pacific Rim International Symposium on Dependable Computing*, pages 328–337, 2013.

[21] D. Z. L. Tian and L. J. Wei. On the Cox model with time-varying regression coefficient. *Journal of the American Statistical Association*, 100(469):172–183, 2005.

[22] J. F. L. Lawless. Regression methods for Poisson process data. *Journal of the American Statistical Association*, 82(399):808–815, 1987.

[23] P. L. Li, M. Shaw, J. Herbsleb, B. Ray, and P. Santhanam. Empirical evaluation of defect projection models for widely-deployed production software systems. In *Proc. 12th ACM SIGSOFT Sympo. on Foundations of Software Eng.*, pages 263–272. ACM, 2004.

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

[25] T. J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320, 1976.

[26] S. Murphy and P. Sen. Time-dependent coefficients in a Cox type regression model. *Stochastic Processes and Their Applications*, 39(1):153–180, 1991.

[27] J. D. Musa. *Software Reliability Data, Technical Report, Data and Analysis Center for Software*. Rome Air Development Center, New York, 1979.

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

[29] Y. Nishio and T. Dohi. Determination of the optimal software release time based on proportional hazards software reliability growth models. *Journal of Quality in Maintenance Engineering*, 9(1):48–65, 2003.

[30] M. Ohba. Inflection s-shaped software reliability growth model. In *Stochastic Models in Reliability Theory*, pages 144–162. Springer, 1984.

[31] K. Ohishi, H. Okamura, and T. Dohi. Gompertz software reliability model: estimation algorithm and empirical validation. *Journal of Systems and Software*, 82(3):535–543, 2009.

[32] H. Okamura and T. Dohi. SRATS: software reliability assessment tool on spreadsheet (Experience report). In *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE 2013)*, pages 100–107, 2013.

[33] H. Okamura, T. Dohi, and S. Osaki. Software reliability growth models with normal failure time distributions. *Reliability Engineering & System Safety*, 116:135–141, 2013.

[34] H. Okamura, Y. Etani, and T. Dohi. A multi-factor software reliability model based on logistic regression. In *2010 IEEE 21st International Symposium on Software Reliability Engineering*, pages 31–40, 2010.

[35] H. Pham. *Software Reliability*. Springer-Verlag, London, 2000.

[36] K. Pillai and V. S. S. Nair. A model for software development effort and cost estimation. *IEEE Transactions on Software Engineering*, 23(8):485–497, 1997.

[37] L. H. Putnam. A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering*, SE-4(4):345–367, 1978.

[38] K. Rinsaka, K. Shibata, and T. Dohi. Proportional intensity-based software reliability modeling with time-dependent metrics. In *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, volume 1, pages 369–376, 2006.

[39] N. F. Schneidewind. Measuring and evaluating maintenance process using reliability, risk, and test metrics. *IEEE Transactions on Software Engineering*, 25(6):768–781, 1999.

[40] N. F. Schnieidewind. Software metrics model for integrating quality control and prediction. In *Proc. 8th Int'l Sympo. on Software Reliab. Eng.*, pages 402–415, 1997.

[41] K. Shibata, K. Rinsaka, and T. Dohi. Metrics-based software reliability models using non-homogeneous poisson processes. In *2006 17th International Symposium on Software Reliability Engineering*, pages 52–61, 2006.

[42] M. Takahashi and Y. Kamayachi. An empirical study of a model for program error prediction. In *Proc. 8th Int'l Conf. on Software Eng.*, pages 330–336. ACM/IEEE CS Press, 1985.

[43] S. Yamada, M. Ohba, and S. Osaki. S-shaped reliability growth modeling for software error detection. *IEEE Transactions on Reliability*, R-32(5):475–478, 1983.

[44] M. Zhao and M. Xie. On maximum likelihood estimation for a general non-homogeneous Poisson process. *Scandinavian Journal of Statistics*, 23(4):597–607, 1996.