A Novel Framework of Software Reliability Evaluation with Software Reliability Growth Models and Software Metrics

Hiroyuki Okamura and Tadashi Dohi
Department of Information Engineering
Graduate School of Engineering, Hiroshima University
1–4–1 Kagamiyama, Higashi-Hiroshima 739–8527 Japan
Email: {okamu,dohi}@rel.hiroshima-u.ac.jp

Abstract—This paper proposes a novel framework of software reliability growth models with software metrics. Our approach is to integrate a classical Poisson-regression-based fault prediction with non-homogeneous Poisson process based software reliability growth models. The remarkable feature of this approach is to handle time series data of fault detections and software metrics for a number of modules at the same time. In the paper, we present the modeling framework that combines Poisson-regression-based fault prediction and software reliability growth models, and also develop an efficient algorithm to estimate model parameters based on EM (expectation-maximization) algorithm. In numerical experiments, by comparing the proposed model with both Poisson-regression-based fault prediction and non-homogeneous Poisson process based software reliability growth models, we discuss the effectiveness of using time series data of fault detections and software metrics from both viewpoints of reliability estimation and fault prediction.

I. INTRODUCTION

Software reliability assessment is an important issue to release high-assurance software products to users/market. During the last four decades, many researches have proposed software reliability growth models (SRGMs) to assess quantitative software reliability from fault data observed in software testing phase [1]–[4]. One of the software reliability assessment activities is to estimate efficiently the number of residual faults causing software failures in testing phase. The accurate estimation for the number of residual faults is quite useful for supporting software testing activities. For example, the number of residual faults may be utilized to determine software release timing to the market.

A commonly-used method to estimate the number of residual faults is based on SRGMs. The SRGM can predict several software reliability measures as well as the number of residual faults from software fault data collections consisting of fault detection time or the number of detected faults. The SRGM is essentially a stochastic model that represents fault detection and removal processes in testing phase. In particular, non-homogeneous Poisson process (NHPP) based SRGMs are quite popular due to their mathematical tractability, and there have been a number of NHPP-based SRGMs proposed by many researchers. However, most of NHPP-based SRGMs focus only on the events of fault detection and removal, i.e., when we estimate model parameters of NHPP-based SRGMs. Although this property makes it easy to handle the NHPP-based SRGMs, it is pointed out that the estimated parameters are not always accurate due to lack of statistical information.

Apart from SRGMs, statistical fault prediction is also significant to develop the high-assurance system. The fault prediction is to know and predict the information on faults in software modules with statistical approaches; how many faults are in modules, what kinds of faults they are, where they are and when they are discovered. Unlike subjective-based approaches, the statistical approaches are based on the objective data called software metrics which are measured from activities in development processes and software products. Many researches studied the fault prediction using software metrics. Almost all the statistical methods are based on the multivariate regression analysis and its variants; linear discriminant analysis [5] and logistic regression [6]. Khoshgoftaar and Munson [7], Khoshgoftaar et al. [8], [9], Evanco and Locovara [10] and Evanco [11] discussed how to use source code metrics such as complexity and coupling of modules toward the software fault prediction. Also, Khoshgoftaar et al. [12] discussed Poisson-regression-based fault prediction.

This paper is to develop a framework of software reliability estimation by combining the traditional NHPP-based SRGMs and Poisson-regression-based fault prediction. As mentioned before, it was pointed out the limitation of NHPP-based SRGMs that use only the information on fault detection data. To overcome this problem, we utilize the information on software metrics to estimate the software reliability measures. In recent years, several papers have discussed the possibility to handle software metrics in the NHPP-based SRGMs [13]–[15]. However, the presented framework of this paper is quite different from the existing metrics-based SRGMs.

The idea behind our framework is to incorporate the Poisson-regression-based fault prediction into the framework of NHPP-based SRGMs. Although the idea is simple, no one has discussed such a mathematical framework with a rigorous statistical approach. There are two advantages of this framework. One advantage is that the framework resolves the statistical problem of NHPP-based SRGMs. In the existing
NHPP-based SRGMs, it is difficult to obtain large samples (fault detection data) to provide accurate estimates of model parameters from only one project. In our framework, it is possible to treat the fault detection data for many projects in only one statistical model. From the statistical point of view, this is a clear advantage to enhance the estimation accuracy of software reliability. Another advantage is to develop the efficient model parameter estimation algorithm. In fact, this paper develops an EM algorithm for the presented framework. The proposed estimation algorithm consists of two parameter estimation algorithms for the ordinary Poisson regression and NHPP-based SRGMs, and thus we can easily implement the estimation algorithms for the ordinary Poisson regression and NHPP-based SRGMs. In numerical experiments, by comparing the proposed model with NHPP-based SRGMs and Poisson-regression-based fault prediction, we discuss the effectiveness of using both software metrics and fault detection data.

II. EARLIER MODELS

A. Poisson-regression-based fault prediction model

Khoshgoftaar et al. [12] discussed prediction of the number of faults in a module by using Poisson regression and zero-inflated Poisson regression models. The Poisson regression model is categorized to the generalized linear model (GLM). The response variable \( Y \) is a Poisson random variable depending on explanatory variables. Let \( X \) be a column vector of multivariate explanatory variables. The probability mass function (p.m.f.) of the response variable \( Y \), which is a non-negative integer, is given by

\[
P(Y = y \mid X = x) = \frac{\mu(x)^y e^{-\mu(x)}}{y!}, \quad y = 0, 1, \ldots, \tag{1}
\]

where \( \mu(x) \) is the mean of the response variable \( Y \) and a function of explanatory variables. In general, samples of explanatory variables are called covariates. In the framework of Poisson regression, it is supposed that

\[
\log \mu(x) = \beta_0 + \beta x, \tag{2}
\]

where \( \beta_0 \) and \( \beta \) are an intercept and a row vector of coefficients for respective covariates, respectively. They are called regression coefficients. Eq. (2) essentially indicates that the Poisson regression is the GLM with a log link function.

In the Poisson regression, we wish to estimate \( (\beta_0, \beta) \) from observed pairs of a response variable and explanatory variables, i.e., \( (y_1, x_1), \ldots, (y_m, \beta_m) \). In general, the zero-inflated Poisson regression deals with the estimation problem when the observed data contain \( y_i = 0 \) for some \( i \). Khoshgoftaar et al. [12] applied the software metrics such as the number of inspections and lines of code to the covariates of Poisson and zero-inflated Poisson regression, and discussed the causal relationship between the number of faults in a module and software metrics. The commonly-used method to estimate regression coefficients in Poisson regression is the maximum likelihood (ML) estimation. Concretely, let \( D = \{(y_1, x_1), \ldots, (y_m, \beta_m)\} \) be observed data where a response variable is the number of faults and covariates are software metrics. Then the log-likelihood function is given by

\[
\mathcal{L}(\beta_0, \beta \mid D) = \sum_{i=1}^{m} (y_i \log \mu(x_i) - \mu(x_i) - \log y_i!). \tag{3}
\]

The ML estimates (MLEs) of \( \beta_0 \) and \( \beta \) can be obtained as the points maximizing the above log-likelihood function. In the GLM framework, the MLEs of regression coefficients are efficiently computed by the iteratively reweighted least squares (IRLS) algorithm. Once the regression coefficients are determined from the observed data, we can predict the number of faults by substituting measured software metrics to Eq. (2).

B. Software reliability growth model

A software reliability growth model (SRGM) is a stochastic model that represents the number of faults in software testing, and is originally developed to evaluate the quantitative software reliability which is the probability that no failure occurs during a fixed time interval [2]. Also, since the SRGM can evaluate the maturity of software testing by estimating the number of remaining faults, SRGMs are often used to make a decision of software release.

SRGMs can be categorized into several classes according to their mathematical backgrounds. In this paper, we focus on the SRGM described by non-homogeneous Poisson processes (NHPPs) which are most popular due to their mathematical tractability. The NHPP-based SRGMs also present the dynamic behavior of the number of detected faults in software testing phase.

Langberg and Singpurwalla [16] presented a modeling framework which contains almost all the NHPP-based SRGMs under the following assumptions:

\[(A-1)\] The number of total software faults in software is a Poisson random variable \( N \) with mean \( \omega \).

\[(A-2)\] All of the fault detections occur at independent and identically distributed (IID) random times, which have a cumulative distribution function (c.d.f.) \( F(t; \theta) \) where \( \theta \) is a parameter vector.

Let \( \{X(t); t \geq 0\} \) be the cumulative number of software faults before time \( t \). Provided that the number of total software faults is given by \( N = n \), the conditional p.m.f. of \( X(t) \) is obtained as the following binomial p.m.f.

\[
P(X(t) = x \mid N = n) = \binom{n}{x} F(t; \theta)^x \overline{F}(t; \theta)^{n-x},
\]

\[(x = 0, 1, \ldots, n), \tag{4}\]

where \( \overline{F}(t; \theta) = 1 - F(t; \theta) \). Since \( N \) is a Poisson random variable with mean \( \omega \), the unconditional p.m.f. of \( X(t) \) is given by

\[
P(X(t) = x) = \sum_{n=x}^{\infty} e^{-\omega} \frac{\omega^n}{n!} \binom{n}{x} F(t; \theta)^x \overline{F}(t; \theta)^{n-x},
\]

\[= \frac{(\omega F(t; \theta))^x}{x!} e^{-\omega F(t; \theta)}, \quad x = 0, 1, \ldots \tag{5}\]
TABLE I
RELATIONSHIP BETWEEN NHPP-BASED SRGMs AND FAULT-DETECTION TIME DISTRIBUTIONS.

<table>
<thead>
<tr>
<th>NHPP-based SRGMs</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goel and Okumoto model [17]</td>
<td>Exponential</td>
</tr>
<tr>
<td>Gamma model [18]</td>
<td>Gamma</td>
</tr>
<tr>
<td>Modified Duane model [19]</td>
<td>Pareto</td>
</tr>
<tr>
<td>Truncated-normal model [20]</td>
<td>Normal</td>
</tr>
<tr>
<td>Log-normal model [20, 21]</td>
<td>Log-normal</td>
</tr>
<tr>
<td>Infection S-shaped model [22]</td>
<td>Truncated-logistic</td>
</tr>
<tr>
<td>Log-logistic model [23]</td>
<td>Log-logistic</td>
</tr>
<tr>
<td>Frechet model [24]</td>
<td>Log-extreme-value-max</td>
</tr>
<tr>
<td>Gompertz model [24]</td>
<td>Truncated-extreme-value-min</td>
</tr>
</tbody>
</table>

Note that $\omega F(t)$ is called a mean value function that represents the expected number of software faults before time $t$. Eq. (5) is exactly same as the p.m.f. of NHPP with mean value function $\omega F(t)$. This implies that the expected number of faults converges to the expected number of total faults $\omega$ eventually. Moreover, NHPP-based SRGMs are characterized by their fault-detection time distribution $F(t)$, i.e., a pattern of the number of detected faults is determined by the fault-detection time distribution $F(t)$. Table I presents typical NHPP-based SRGMs and their fault-detection time distributions.

In NHPP-based SRGMs, model parameters are estimated from fault detection data, i.e., a time series of the number of detected faults in testing phase. Let $D = \{(t_1, x_1), \ldots, (t_K, x_K)\}$ be the fault detection data, where $(t_k, x_k)$ represents a pair of interval time of the $k$-th testing and the number of detected faults in the $k$-th testing. The ML estimation is commonly used in NHPP-based SRGMs. The log-likelihood function of NHPP-based SRGMs is generally given by

$$
\mathcal{L}(\omega, \theta) = \sum_{k=1}^{K} n_k \log \omega + \sum_{k=1}^{K} n_k \log (F(u_k; \theta) - F(u_{k-1}; \theta)) - \sum_{k=1}^{K} \log n_k! - \omega F(u_K; \theta),
$$

(6)

where $u_k = \sum_{l=1}^{k} t_l$. The MLE of NHPP-based SRGM can be obtained by maximizing the above log-likelihood function. Since Eq. (6) is a non-linear function with respect to $\omega$ and $\theta$, any numerical technique should be applied to find the MLE. In [26], Okamura et al. presented the EM (expectation-maximization) algorithm for NHPP-based SRGMs. Compared to general-purpose maximization techniques such as Newton’s method and Nelder-Mead simplex method, the EM algorithm can stably provide MLEs for any types of fault detection data.

III. NHPP-BASED SRGM WITH POISSON REGRESSION

A. Model description

This paper presents an integration model of both NHPP-based SRGMs and Poisson-regression-based fault prediction.

Since the number of total faults is assumed to be a Poisson random variable in the modeling framework of NHPP-based SRGMs [16], we can apply the regression model to represent the expected number of total faults according to Poisson regression scheme.

Concretely, we suppose that there are $m$ modules in software, and their software metrics vectors are given by $x_1, \ldots, x_m$. Consider the following assumptions:

(B-1) The number of total software faults in the $i$-th software module is a Poisson random variable $N_i$ with mean $\mu(x_i)$;

$$
\log \mu(x_i) = \beta_0 + \beta x_i,
$$

(7)

where $x_i$ is a vector of software metrics of the $i$-th module.

(B-2) All of the fault detections in the $i$-th module occur at independent and IID random times, which have a c.d.f. $F_i(t; \theta)$ where $\theta_i$ is a parameter vector of the fault-detection time distribution.

The essential difference between Langberg and Singpurwalla’s assumptions and our assumptions is that the expected number of total faults is modeled by the regression with software metrics.

Let $\{X_i(t); t \geq 0\}$ be the cumulative number of faults detected before testing time $t$. Similar to the ordinary NHPP-based SRGM, the p.m.f. of $X(t)$ is given by

$$
P(X_i(t) = x) = \frac{(\mu(x_i)F_i(t; \theta_i))^x e^{-\mu(x_i)F_i(t; \theta_i)}}{x!},
$$

(8)

Note that the above p.m.f. indicates that the cumulative number of detected faults in the $i$-th module is also an NHPP and that the mean value function depends on the software metrics of the $i$-th module. In this paper, the presented model is called PR-NHPP model.

B. Parameter estimation

To evaluate the software reliability using PR-NHPP model, model parameters should be estimated so that the model is fitted to observed data. This section presents an efficient parameter estimation algorithm for PR-NHPP model. The point of our approach is to ‘simultaneously’ estimate all the NHPP parameters and Poisson regression coefficients by using a likelihood function. It should be noted that our estimates are different from estimates when the Poisson regression coefficients are estimated independently before NHPP parameter estimation. That is, in our approach, the estimates of Poisson regression coefficients are affected by the estimates of NHPP parameters and vice versa. In addition, we propose a simple iterative estimation to estimate all the NHPP parameters and Poisson regression coefficients simultaneously.

Suppose that there are $m$ software modules and that their software metrics vectors $x_1, \ldots, x_m$ are measured. Let $D_i = \{(t_{i,1}, n_{i,1}), \ldots, (t_{i,K_i}, n_{i,K_i})\}$ be the fault detection data for the $i$-th module, i.e., a time series of the number of faults.
detected in the $i$-th module, where $(t_{i,k}, n_{i,k})$ denotes a pair of interval time of the $k$-th testing and the number of faults detected by the $k$-th testing in $i$-th module.

For given observed data $x_1, \ldots, x_m$ and $D_1, \ldots, D_m$, the log-likelihood function is formulated as follows.

$$
L(\beta_0, \beta, \theta_1, \ldots, \theta_m) = 
\sum_{i=1}^{m} \sum_{k=1}^{K_i} n_{i,k} \log \mu(x_i) + \sum_{i=1}^{m} \sum_{k=1}^{K_i} n_{i,k} \log \left(F_i(u_{i,k}; \theta_i) - F_i(u_{i,k-1}; \theta_i)\right)
- \sum_{i=1}^{m} \sum_{k=1}^{K_i} \log n_{i,k}! - \sum_{i=1}^{m} \mu(x_i)F_i(u_{i,K_i}; \theta_i),
$$  

(9)

where $u_{i,k} = \sum_{j=1}^{k} t_{i,j}$. Then the MLEs of PR-NHPP model are the parameters maximizing the above log-likelihood function. Since Eq. (9) is non-linear, general-purpose maximization approaches do not work well to get the MLE. Therefore, in this paper, we consider an EM algorithm for PR-NHPP model estimation.

The EM algorithm is an iterative method for computing ML estimates with incomplete data [27]. In general, let $D$ and $U$ be observable and unobservable data, respectively, and we wish to estimate a model parameter vector $\lambda$ from the observable data $D$.

The EM algorithm consists of E-step and M-step. E-step computes the expectation of log-likelihood function of the complete data $(D, U)$ with the distribution for unobservable data when a provisional parameter vector $\lambda'$ is given. Formally, the expected log-likelihood function to be computed in the E-step can be written as

$$
Q(\lambda | \lambda') = \int p(U | D; \lambda') \log p(D, U | \lambda) dU,
$$  

(10)

where $p(\cdot | \cdot)$ is any probability density or mass function. In the above equation, $p(U | D; \lambda')$ and $p(D, U | \lambda)$ correspond to the p.d.f. of unobserved data and the complete likelihood function, respectively. Furthermore, the p.d.f. of unobserved data can be derived from Bayes theorem:

$$
p(U | D; \lambda') = \frac{p(D, U; \lambda')}{p(D; \lambda')}.  
$$  

(11)

On the other hand, in the M-step, we find a new parameter vector $\lambda''$ that maximizes the expected log-likelihood function, i.e.,

$$
\lambda'' := \arg \max_{\lambda} Q(\lambda | \lambda').
$$  

(12)

The parameter vector $\lambda'$ becomes provisional parameters in the next E- and M-steps. The one EM step updates model parameters and ensures that the log-likelihood function increases. The EM steps are repeatedly executed until model parameters converges.

Consider the EM algorithm for PR-NHPP model. We define the unobserved data $U$ in the PR-NHPP model as a set of the number of total faults $N_1, \ldots, N_m$ and the exact fault-detection time sequence for all the faults $S_{i,1} < S_{i,2} < \cdots < S_{i,N_i}$ for all $i = 1, \ldots, m$. Obviously, since these time sequences include fault-detection time in future, they are unobserved data. According to the order statistics argument, the likelihood function of the complete data can be written as

$$
p(D, U; \beta_0, \beta, \theta_1, \ldots, \theta_m) = \prod_{i=1}^{m} \left(\mu(x_i)\, e^{-\mu(x_i)} \prod_{k=1}^{N_i} f_i(S_{i,k}; \theta_i)\right),
$$  

(13)

where $\lambda = (\beta_0, \beta, \theta_1, \ldots, \theta_m)$, $\lambda' = (\beta_0', \beta', \theta_1', \ldots, \theta_m')$ and $f_i(\cdot; \theta_i)$ is the p.d.f. of $F_i(\cdot; \theta_i)$. By taking the expected value under the distribution of unobserved data with a provisional parameter $\lambda' = (\beta_0', \beta', \theta_1', \ldots, \theta_m')$, we have

$$
Q(\lambda | \lambda') = \sum_{i=1}^{m} E_{\lambda'}(N_i) \log \mu(x_i) - E_{\lambda'} \left[\sum_{i=1}^{m} \sum_{k=1}^{N_i} \log f_i(S_{i,k}; \theta_i)\right],
$$  

(14)

where $E_{\lambda'}[\cdot]$ indicates the expectation with the provisional parameter $\lambda'$. It should be noted that only the first two terms in the right-hand side of Eq. (14) contain the regression coefficients $\beta_0$ and $\beta$ to be computed, and that the third term includes the parameters of fault-detection time distribution $\theta_1, \ldots, \theta_m$. Therefore, the M-step of PR-NHPP model can be divided into two procedures; finding the regression coefficients that maximize the first two terms and finding the distribution parameters that maximize the third term.

By comparing the first two terms in Eq. (14) with Eq. (3), we find that the regression coefficients in the M-step of PR-NHPP model can be computed by solving the ordinary Poisson regression whose response variables are the expected values $E_{\lambda'}(N_i), i = 1, \ldots, m$. According to the result in [26], the expected number of total faults in the $i$-th module under given provisional parameters $\beta_0', \beta', \theta_1', \ldots, \theta_m'$ can be computed as

$$
E_{\lambda'}(N_i) = \sum_{k=1}^{K_i} n_k + e^{\beta_0' + \beta' x_i} P_i(s_{K_i}; \theta_i').
$$  

(15)

Also, the distribution parameters $\theta_i$ can be obtained by the following maximization problem:

$$
\theta_i' = \arg \max_{\theta_i} E_{\lambda'} \left[\sum_{k=1}^{N_i} \log f_i(S_{i,k}; \theta_i)\right].
$$  

(16)

This maximization procedure is reduced to the EM-step for the ordinary NHPP-based SRGM [26] by substituting $e^{\beta_0' + \beta' x_i}$ into the provisional value of the number of total faults. Finally, the EM-step for PR-NHPP model is given in Algorithm 1. In the algorithm, $\operatorname{PR}(y_1, \ldots, y_m, x_1, \ldots, x_m)$ indicates the ordinary Poisson regression to obtain the regression coefficients with explanatory variables $x_1, \ldots, x_m$ and response variables $y_1, \ldots, y_m$. Also, $\operatorname{EM}(\omega, \theta, D, F')$ means the EM-step for the ordinary NHPP-based SRGM whose fault-detection time distribution is $F(t; \theta)$ under the observed fault
detection data $\mathcal{D}$ to update parameters $\omega$ and $\theta$. Although EM($\omega, \theta; \mathcal{D}; F$) updates the parameter $\omega$, the parameter $\omega$ is ignored in this algorithm. The proposed EM algorithm to get the MLE for PR-NHPP model is quite simple for implementation, because it is only necessary to invoke two procedures; the ordinary Poisson regression routine and the EM procedure for the ordinary NHPP-based SRGMs. Moreover, the algorithm is stable to get the MLE, compared to general-purpose maximization algorithms.

**Algorithm 1** EM-step for PR-NHPP model.

<table>
<thead>
<tr>
<th>Regression coefficients (input and output): $\beta_0, \beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution parameters (input and output): $\theta_1, \ldots, \theta_m$</td>
</tr>
<tr>
<td>Module metrics (input): $x_1, \ldots, x_m$</td>
</tr>
<tr>
<td>Fault detection data (input): $\mathcal{D}_1, \ldots, \mathcal{D}_m$</td>
</tr>
</tbody>
</table>

1: for $i = 1 : m$ do
2: \quad $\omega_i \leftarrow \exp(\hat{\beta}_0 + \beta x_i)$
3: \quad $N_i \leftarrow \sum_{k=1}^{K_i} n_{i,k} + \omega_i \bar{F}_i(s_{i,K_i}; \hat{\theta}_i)$
4: \quad $\hat{\theta}_i \leftarrow \text{EM}(\omega_i, \theta_i; \mathcal{D}_i; F_i)$
5: end for
6: $(\hat{\beta}_0, \hat{\beta}) \leftarrow \text{PR}(N_1, \ldots, N_m, x_1, \ldots, x_m)$

D. Reliability measure

After getting MLEs of PR-NHPP model parameters, we can derive several measures with respect to software reliability for a module. This paper presents typical measures for the software reliability; software reliability function, the expected number of total faults and fault-free probability. The software reliability function is defined by a probability that any fault is detected in time period $[s, s + t]$. Let $\beta_0$, $\beta$ and $\theta_i$ be MLEs for the regression coefficients and distribution parameters. The estimated number of detected faults for the $i$-th module obeys the NHPP with mean value function $e^{\hat{\beta}_0 + \hat{\beta} x_i} F_i(t; \hat{\theta}_i)$. Thus the software reliability function is given by the following conditional probability:

$$R_i(t+s|s) = \exp \left( -e^{\hat{\beta}_0 + \hat{\beta} x_i} (F_i(t+s; \hat{\theta}_i) - F_i(s; \hat{\theta}_i)) \right).$$

Also, as shown in Section III-B, the distribution of unobserved data can be derived from Bayes theorem under given model parameters. The p.m.f. of the number of total faults in the $i$-th module is given by

$$P(N_i = x|\mathcal{D}) = \frac{\left( \hat{\omega}_i \bar{F}_i(s_{i,K_i}; \hat{\theta}_i) \right)^{x - \sum_{k=1}^{K_i} n_{i,k}}}{(x - \sum_{k=1}^{K_i} n_{i,k})!} e^{-\hat{\omega}_i \bar{F}_i(s_{i,K_i}; \hat{\theta}_i)},$$

where $\hat{\omega}_i = e^{\hat{\beta}_0 + \hat{\beta} x_i}$. Then the predictive number of total faults (TF) for the $i$-th module is

$$\text{TF}_i = \sum_{k=1}^{K_i} n_{i,k} + e^{\hat{\beta}_0 + \hat{\beta} x_i} \bar{F}_i(s_i; \hat{\theta}_i).$$

The number of residual faults can be computed by $\sum_{k=1}^{K_i} n_{i,k}$. From the number of total faults. Moreover, since the number of total faults follows the Poisson distribution, the probability that the $i$-th module has no fault (fault-free probability: FFP) is given by

$$\text{FFP}_i = \exp \left( -e^{\hat{\beta}_0 + \hat{\beta} x_i} \bar{F}_i(s_i; \hat{\theta}_i) \right).$$

## IV. EXPERIMENTS

### A. Reliability estimation

This section investigates the effectiveness of PR-NHPP model in terms of both reliability estimation and fault prediction by comparing it with Poisson-regression-based prediction model and NHPP-based SRGMs. In the experiment, we focus on the open source project; Apache Tomcat\(^1\). Apache Tomcat provides Java Servlets and their components are written by Java and C languages. The experiments deal with the versions of Apache Tomcat 5.5.12 through 5.5.36, since we choose fully mature application. The fault detection data are

\(^1\)http://tomcat.apache.org/
The selected regression coefficients are LOC, St and Ac and their estimates are $\beta_0 = 2.57$, $\beta_{LOC} = 5.72e^{-5}$, $\beta_{St} = -1.45e -4$ and $\beta_{Ac} = 0.532$. Also Table V shows the estimated parameters of fault detection time distribution in the ordinary NHPP-based SRGMs (NHPP) and PR-NHPP model. To investigate only the difference of the estimated values in NHPP and PR-NHPP, we omit the concrete p.d.f.'s for the fault detection time distribution. Notice that the fault detection time distributions except for Tester have two parameters. From these results, we find that the estimated parameters of fault detection time distribution in PR-NHPP is almost same ones estimated in the ordinary NHPP-based SRGMs. Table VI presents the reliability measures; the predictive number of total faults (TF) and the fault-free probability (FFP) for each module. Compared to the difference of estimates of the fault detection time distribution, RF and FFP are affected by the metrics. Also the predictive number of total faults in PR-NHPP model are little improved, except for the cases of Servlets and Tester, in terms of the difference from the number of detected faults in July 2012.

### B. Fault prediction

In the next experiment, we conduct the prediction of the number of total faults, i.e., the number of faults detected until July 2012, based on two methods; Poisson regression with only code metrics and PR-NHPP with both code metrics and fault detection data. The experiment is performed according to leave-one-out cross validation. In this scheme, the data of

---

**TABLE II**

<table>
<thead>
<tr>
<th>Module</th>
<th>Faults 7/2009</th>
<th>Faults 7/2012</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalina</td>
<td>237</td>
<td>272</td>
<td>The servlet container core</td>
</tr>
<tr>
<td>Connector</td>
<td>75</td>
<td>89</td>
<td>Coyote connectors</td>
</tr>
<tr>
<td>Jasper</td>
<td>70</td>
<td>74</td>
<td>The JSP (JavaServer Pages) compiler and runtime engine</td>
</tr>
<tr>
<td>Servlets</td>
<td>51</td>
<td>57</td>
<td>Servlet API and support programs for CGI, SSI and WebDAV</td>
</tr>
<tr>
<td>Tester</td>
<td>1</td>
<td>1</td>
<td>Unit testing framework</td>
</tr>
<tr>
<td>Webapps</td>
<td>53</td>
<td>65</td>
<td>Web application for administration, documentation and examples</td>
</tr>
</tbody>
</table>

**TABLE III**

<table>
<thead>
<tr>
<th>Module</th>
<th>LOC</th>
<th>St</th>
<th>Fn</th>
<th>Ac</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalina</td>
<td>125462</td>
<td>39386</td>
<td>4543</td>
<td>2.79</td>
</tr>
<tr>
<td>Connector</td>
<td>168482</td>
<td>65829</td>
<td>5912</td>
<td>3.37</td>
</tr>
<tr>
<td>Jasper</td>
<td>41861</td>
<td>15557</td>
<td>1691</td>
<td>2.88</td>
</tr>
<tr>
<td>Servlets</td>
<td>15480</td>
<td>1944</td>
<td>400</td>
<td>1.45</td>
</tr>
<tr>
<td>Tester</td>
<td>12677</td>
<td>4750</td>
<td>409</td>
<td>3.36</td>
</tr>
<tr>
<td>Webapps</td>
<td>37782</td>
<td>12851</td>
<td>1035</td>
<td>2.32</td>
</tr>
</tbody>
</table>

**TABLE IV**

<table>
<thead>
<tr>
<th>Module</th>
<th>NHPP-based SRGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalina</td>
<td>Log-normal model</td>
</tr>
<tr>
<td>Connector</td>
<td>Frechet model</td>
</tr>
<tr>
<td>Jasper</td>
<td>Log-logistic model</td>
</tr>
<tr>
<td>Servlets</td>
<td>Frechet model</td>
</tr>
<tr>
<td>Tester</td>
<td>Goel and Okamoto model</td>
</tr>
<tr>
<td>Webapps</td>
<td>Infection S-shaped model</td>
</tr>
</tbody>
</table>

**TABLE V**

<table>
<thead>
<tr>
<th>Module</th>
<th>Distribution</th>
<th>NHPP</th>
<th>PR-NHPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalina</td>
<td>Log-normal</td>
<td>(0.44, 6.93)</td>
<td>(0.44, 6.93)</td>
</tr>
<tr>
<td>Connector</td>
<td>Log-extreme-value-max</td>
<td>(0.45, 6.82)</td>
<td>(0.45, 6.82)</td>
</tr>
<tr>
<td>Jasper</td>
<td>Log-logistic</td>
<td>(0.27, 6.85)</td>
<td>(0.26, 6.84)</td>
</tr>
<tr>
<td>Servlets</td>
<td>Log-extreme-value-max</td>
<td>(0.36, 6.67)</td>
<td>(0.36, 6.66)</td>
</tr>
<tr>
<td>Tester</td>
<td>Exponential</td>
<td>1.62e-5</td>
<td>6.24e-6</td>
</tr>
<tr>
<td>Webapps</td>
<td>Truncated-logistic</td>
<td>(207.7, 853.4)</td>
<td>(210.6, 852.2)</td>
</tr>
</tbody>
</table>

**TABLE VI**

<table>
<thead>
<tr>
<th>Module</th>
<th>TF</th>
<th>FFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalina</td>
<td>254.06</td>
<td>3.88e-8</td>
</tr>
<tr>
<td>Connector</td>
<td>89.69</td>
<td>4.16e-7</td>
</tr>
<tr>
<td>Jasper</td>
<td>74.25</td>
<td>1.43e-2</td>
</tr>
<tr>
<td>Servlets</td>
<td>55.16</td>
<td>1.56e-2</td>
</tr>
<tr>
<td>Tester</td>
<td>31.44</td>
<td>6.04e-14</td>
</tr>
<tr>
<td>Webapps</td>
<td>53.24</td>
<td>7.88e-1</td>
</tr>
</tbody>
</table>

---

The bug reports of Apache Tomcat are classified in terms of software modules. In the experiments, we collected reports for minor, normal and critical bugs detected in 6 modules: Catalina, Connectors, Jasper, Servlet, Tester and Webapps. Each module consists of a lot of Java and C programming files. These bugs were reported in the period from May 2004 until July 2012. Table II summarizes software modules and their total numbers of bugs detected until July 2009 and July 2012, respectively. From this table, almost all bugs have been detected until July 2009 which is an intermediate point between May 2004 and May 2013, and thus all the current modules are regarded to be stable in terms of bug fixing.

Next we measure software metrics for modules. Although design metrics measured from specifications, which were used in [12], are more appropriate in the interest of fault prediction, it is generally difficult to measure such design metrics for the open source projects. Thus we use software metrics regarding program sources; lines of code (LOC), the number of statements (St), the number of functions (Fn) and the average McCabe software complexity (Ac). Table III presents these metrics for all the modules in Tomcat version 5.5.12. Throughout this experiment, we use these metrics.

In the first experiment, we investigate the ability of software reliability estimation of PR-NHPP model by comparing the ordinary NHPP-based SRGMs. Table IV presents the selected NHPP-based SRGMs with the smallest AIC for the fault detection data of July 2009. Based on these models, we estimated the parameters of PR-NHPP model with the metrics data. The selected regression coefficients are LOC, St and Ac and their estimates are $\beta_0 = 2.57$, $\beta_{LOC} = 5.72e^{-5}$, $\beta_{St} = -1.45e -4$ and $\beta_{Ac} = 0.532$. Also Table V shows the estimated parameters of fault detection time distribution in the ordinary NHPP-based SRGMs (NHPP) and PR-NHPP model. To investigate only the difference of the estimated values in NHPP and PR-NHPP, we omit the concrete p.d.f.’s for the fault detection time distribution. Notice that the fault detection time distributions except for Tester have two parameters. From these results, we find that the estimated parameters of fault detection time distribution in PR-NHPP is almost same ones estimated in the ordinary NHPP-based SRGMs. Table VI presents the reliability measures; the predictive number of total faults (TF) and the fault-free probability (FFP) for each module. Compared to the difference of estimates of the fault detection time distribution, RF and FFP are affected by the metrics. Also the predictive number of total faults in PR-NHPP model are little improved, except for the cases of Servlets and Tester, in terms of the difference from the number of detected faults in July 2012.
Table VII shows the results of the fault prediction by Poisson regression (PR) and PR-NHPP model. In the Poisson regression, we apply the variable selection method with AIC. Table VIII presents the estimated regression coefficients in the case of Poisson regression. In the table, blanks indicates variables which have not been selected in the variable selection method. Similarly, Table IX indicates the estimated regression coefficients in the case of PR-NHPP model after the model and variable selection described in Section III-C. Note that the predictive number of faults in Table VII are computed by only the estimated parameters, we predict the number of faults in a module. Moreover, from Table IX, we find that the several regression coefficients of PR-NHPP model were not selected. This is because the information on some software metrics is not needed due to the information gain of fault detection data. However, the accuracy of the prediction is not so high. This is caused by the lack of the numbers of modules and software metrics used in the experiment. This experiment dealt with only 6 modules and 4 metrics. Since the proposed estimation algorithm is scalable with respect to the number of modules, the accuracy can be improved by increasing the number of modules to be used in the fault prediction.

V. Conclusion

This paper has presented a new framework for the software reliability estimation using both software metrics and fault detection data. The developed parameter estimation algorithm has been composed of the parameter estimation procedures for the ordinary Poisson regression and NHPP-based SRGMs. In the numerical experiments, we have exhibited reliability estimation and fault prediction for the Apache Tomcat by using PR-NHPP model. As a result, we find that the information gain of fault detection data is significant to predict the number of faults in a module.

The presented parameter estimation algorithm is easily extended by replacing Poisson regression with other statistical techniques. For example, in the field of empirical software engineering, it is happened that the number of metrics is more than the size of samples when the metrics are microscopically measured from program sources. In such case, it is known that the shrinkage estimation, such as the penalized likelihood estimation, is effective. The advantage of our algorithm is that Poisson regression can easily be replaced with such shrinkage estimation. In future, we will consider measurement of appropriate software metrics to evaluate both software reliability and fault proneness in the presented framework. In addition, we try to extend the estimation algorithm to the shrinkage estimation to deal with microscopic software metrics.

Acknowledgment

This research was supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific research (C), Grant No. 23500047 (2011–2013) and Grant No. 23510171 (2011–2013).

References


<table>
<thead>
<tr>
<th>Module</th>
<th>Faults</th>
<th>Poisson regression</th>
<th>PR-NHPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalina</td>
<td>272</td>
<td>3.14e-16</td>
<td>77.65</td>
</tr>
<tr>
<td>Connector</td>
<td>89</td>
<td>9.38e+5</td>
<td>525.05</td>
</tr>
<tr>
<td>Jasper</td>
<td>74</td>
<td>0.82</td>
<td>157.98</td>
</tr>
<tr>
<td>Servlets</td>
<td>57</td>
<td>3508.73</td>
<td>56.29</td>
</tr>
<tr>
<td>Tester</td>
<td>1</td>
<td>85.83</td>
<td>83.37</td>
</tr>
<tr>
<td>Webapps</td>
<td>65</td>
<td>6.26</td>
<td>66.62</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>(intercept)</th>
<th>LOC</th>
<th>St</th>
<th>Fn</th>
<th>Ac</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalina</td>
<td>41.1</td>
<td>-1.94e-3</td>
<td>4.78e-3</td>
<td>2.81e-3</td>
<td>-12.0</td>
</tr>
<tr>
<td>Connector</td>
<td>10.9</td>
<td>-2.51e-4</td>
<td>6.96e-4</td>
<td>1.89e-3</td>
<td>-3.52</td>
</tr>
<tr>
<td>Jasper</td>
<td>2.96</td>
<td>3.88e-4</td>
<td>-2.90e-4</td>
<td>6.95e-3</td>
<td>-1.09</td>
</tr>
<tr>
<td>Servlets</td>
<td>15.8</td>
<td>1.19e-4</td>
<td>1.18e-4</td>
<td>4.97e-3</td>
<td>-4.75</td>
</tr>
<tr>
<td>Tester</td>
<td>2.67</td>
<td>5.86e-5</td>
<td>-1.49e-4</td>
<td>0.520</td>
<td></td>
</tr>
<tr>
<td>Webapps</td>
<td>7.46</td>
<td>-2.13e-4</td>
<td>6.67e-3</td>
<td>1.92</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>(intercept)</th>
<th>LOC</th>
<th>St</th>
<th>Fn</th>
<th>Ac</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalina</td>
<td>3.65</td>
<td>0.231</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connector</td>
<td>3.71</td>
<td>1.52e-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jasper</td>
<td>2.76</td>
<td>-1.19e-4</td>
<td>1.21e-3</td>
<td>0.730</td>
<td></td>
</tr>
<tr>
<td>Servlets</td>
<td>2.76</td>
<td>5.71e-5</td>
<td>-1.43e-4</td>
<td>0.481</td>
<td></td>
</tr>
<tr>
<td>Tester</td>
<td>2.77</td>
<td>5.69e-5</td>
<td>-1.43e-4</td>
<td>0.481</td>
<td></td>
</tr>
<tr>
<td>Webapps</td>
<td>2.77</td>
<td>5.69e-5</td>
<td>-1.43e-4</td>
<td>0.481</td>
<td></td>
</tr>
</tbody>
</table>


